[Tips | Elemental Selenium]

Install Java JDK, IntelliJ, WebDriver Jars.

Selenium Clients and WebDriver Language Bindings
In order to create scripts that interact with the Selenium Server (Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers.
While language bindings for other languages exist, these are the core ones that are supported by the main project hosted on GitHub.
C#
Stable: 4.1.0 (November 22, 2021)
Changelog
API Docs
Ruby
Stable: 4.1.0 (November 22, 2021)
Changelog
API Docs
Java
Changelog
API Docs
Python
Stable: 4.1.0 (November 22, 2021)
Changelog
API Docs
JS
JavaScript
Stable: 4.1.0 (November 22, 2021)
Changelog
API Docs

Create a folder in C Drive: Selenium.

Download Chrome and Firefox Webdrivers.

We need to know the version of the web browser

Mine is: Version 126.0.6478.127 (Official Build) (64-bit)
[Chrome for Testing availability (googlechromelabs.github.io)]

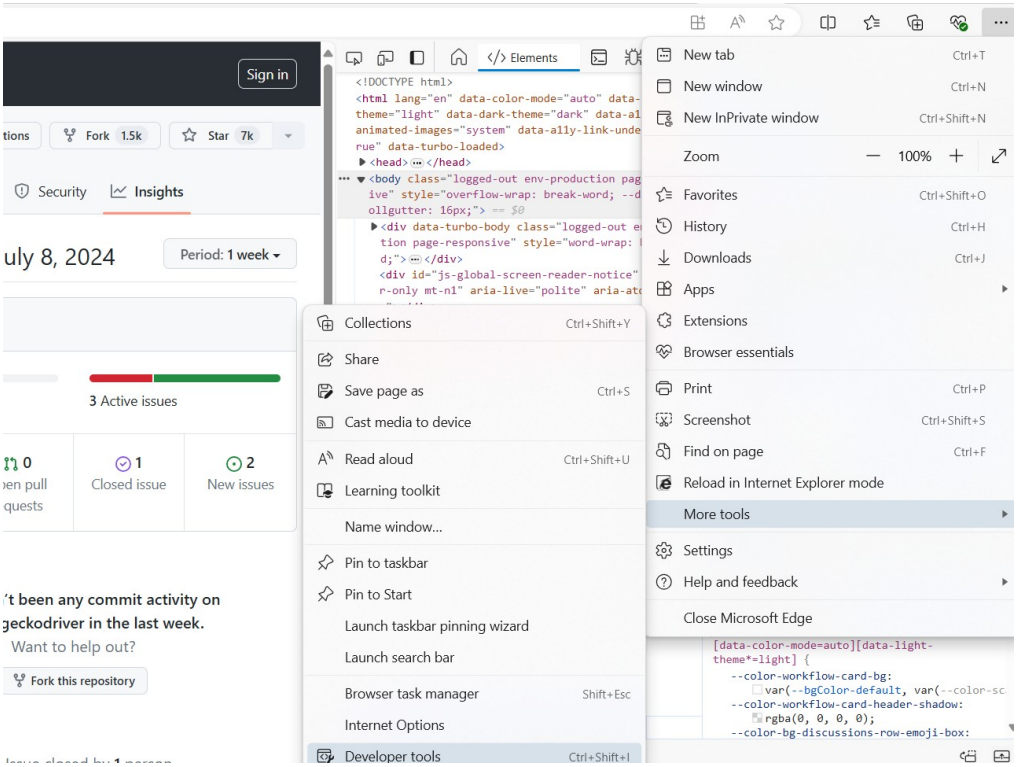
[Releases · mozilla/geckodriver (github.com)]

INSTALLATION
Below steps apply to both Windows and MAC
. Create new Java project in IntelliJ.
Name it: SelProject
. Add Jar files to the project
a. Click go to File -> Project Structure -> in a project setting tab look for Modules -> Dependencies -> Click on '+' Sign -> Select for JARs or directories.
b. Select all the selenium .jar files from the directory and subdirectory /lib, where you have extracted after download.
. Download browser drivers
a. Under download section download Chrome and Firefox drivers.
. Create a resources folder in the project and put all 2 drivers there

Create a class under src folder : MyFirstTest

Needs public static void main(String[] args) {} to be run

We can open element with click button on ... Menu



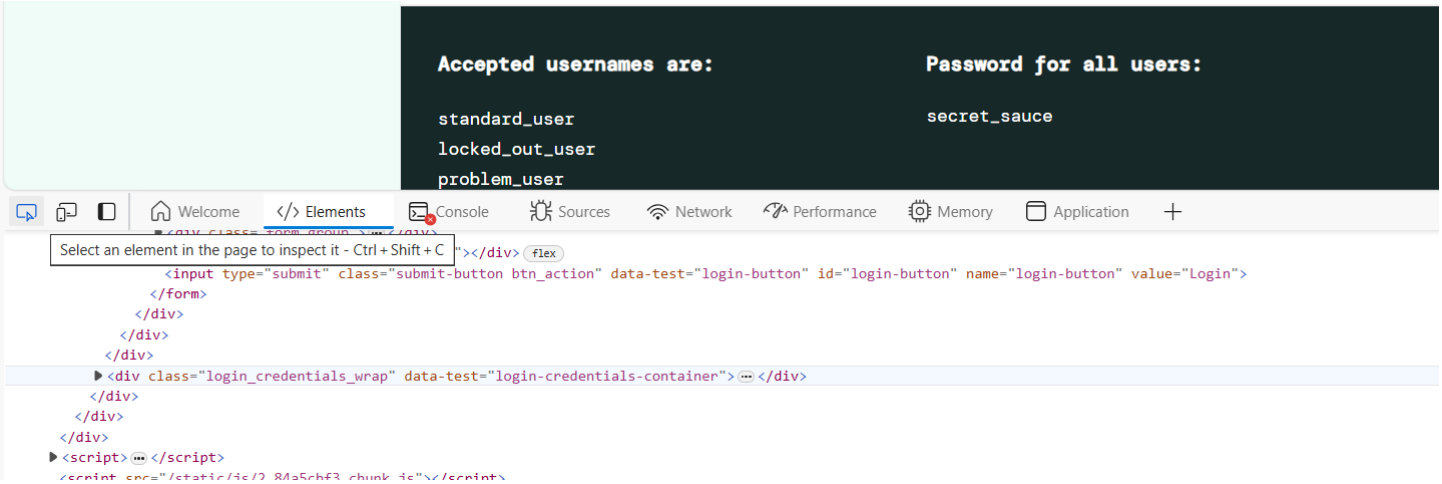
Tips:

- Shortcut to write main method on IntelliJ:

Write : main +Tab :

```
public static void main(String[] args) {

}
```

- Click on the button to inspect a specific element.



- Double check if the ID is unique

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class MyFirstTest {
    public static void main(String[] args) throws InterruptedException {

        System.setProperty("webdriver.chrome.driver", "resources/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
       // driver.navigate().to("https://www.yahoo.com");
       // System.out.println(driver.getTitle());
        //driver.close();
/*
        // Set the path to GeckoDriver executable
        System.setProperty("webdriver.gecko.driver", "resources/geckodriver.exe");

        // Initialize FirefoxDriver
        WebDriver driver1 = new FirefoxDriver();

        driver1.close();*/
//Workaround for MAC users version 115
//Workaround for Chromedriver 115.**.****.102
/*==========================================
        ChromeOptions options = new ChromeOptions();
        options.setBinary("/Applications/Google Chrome.app/Contents/MacOS/Google
Chrome");
*/
//Create String obj with url address value
String urlAddress= "https://www.uber.com";
//Pass string as an argument to GET method, but it waits for all elements to be loaded,
and the history is not saved.
driver.get(urlAddress);

//Maximize the size of the window
driver.manage().window().maximize();

//Navigate.to() is almost same as get(), but it doesn't wait for all elements to be
loaded, and the history is saved.
//before proceeding to the next step.
driver.navigate().to("https://www.yahoo.com");
```

```
Thread.sleep(10000);
//To go back to uber.com we are using navigate().back()
driver.navigate().back();
Thread.sleep(5000);
//To make sure, let's print the title of the uber.com
System.out.println(driver.getTitle());

//Navigate to golds gym
        driver.get("https://www.goldsgym.com/locate-a-gym/?
utm_source=internal&utm_medium=ppc");
//Print the title of the page
        String webSiteTitle = driver.getTitle();
        System.out.println(webSiteTitle);
//Get current URL
        String currentURLAddress=driver.getCurrentUrl();
//To make sure that we are on the correct webpage, we check if the URL contains specific
text
        if(currentURLAddress.contains("locate-a-gym")){
            System.out.println("I'm on the correct page");
        }

//Before we go forward to Yahoo.com we are going to use Thread.sleep
Thread.sleep(5000);
//Navigate forward to yahoo.com
driver.navigate().forward();
//Print the title of yahoo.com
System.out.println(driver.getTitle());
//Get only the current URL
String currentURLAddress2=driver.getCurrentUrl();

//Print current URL address
System.out.println(currentURLAddress2);

//Refresh the page
driver.navigate().refresh();
//Close only the current window:
driver.close();
//Close the Whole pages:
driver.quit();


    }
}
```

WELCOME TO SELENIUM CLASS

Please, fill out the following form:

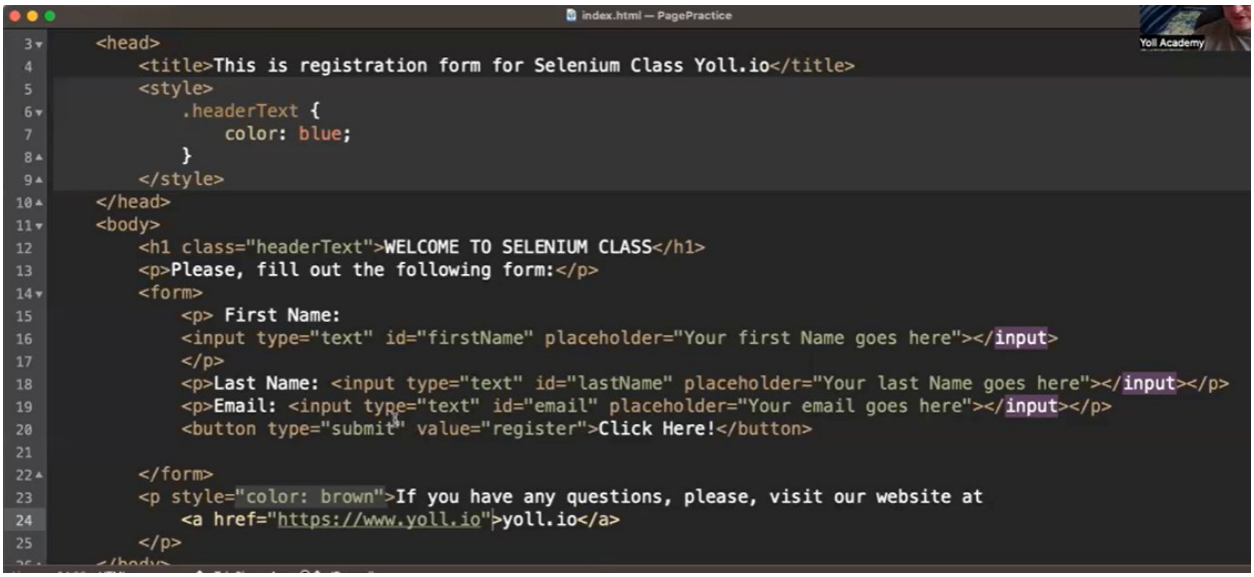
First Name: Your first Name goes her

Last Name: Your last Name goes her

Email: Your email goes here

Click Here!

If you have any questions, please, visit our website at yoll.io



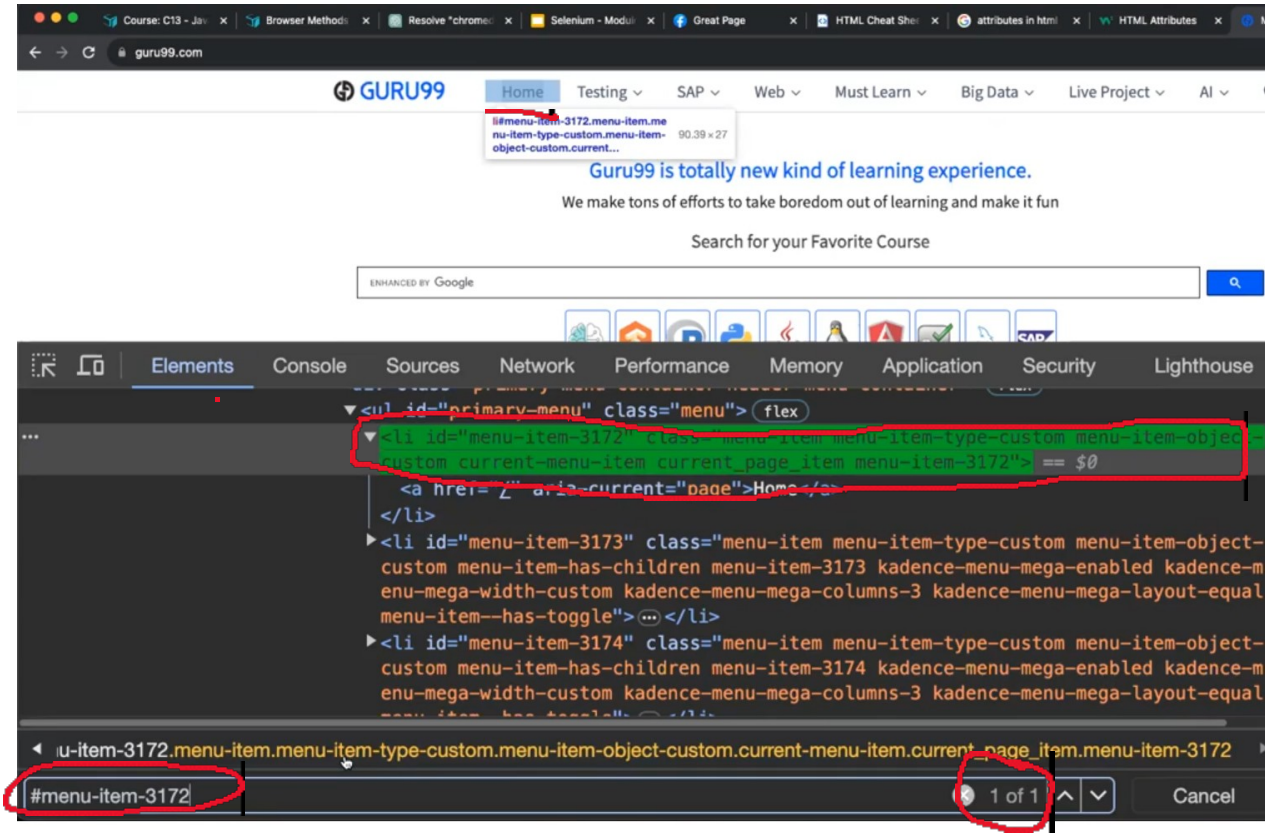
Locators:

By ID, must be the first choice, it supposed to be unique, but can be duplicated.

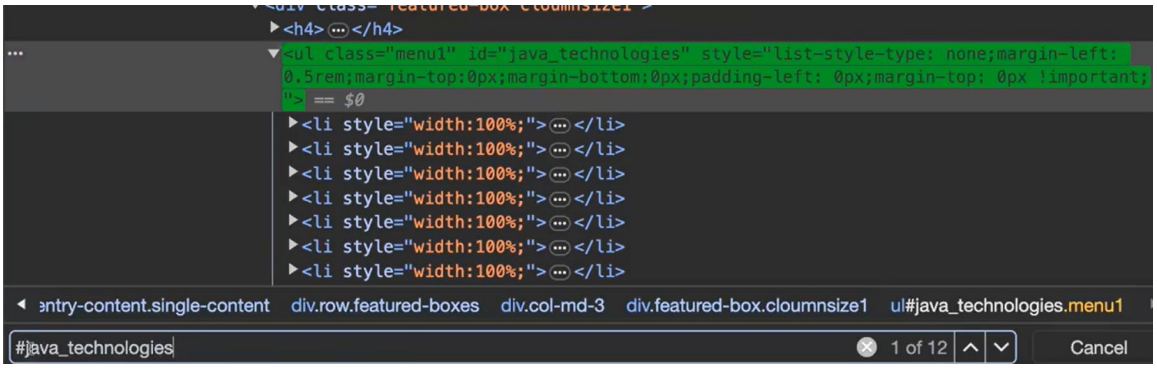
Alt+F to search.

In the picture, we have one ID element.

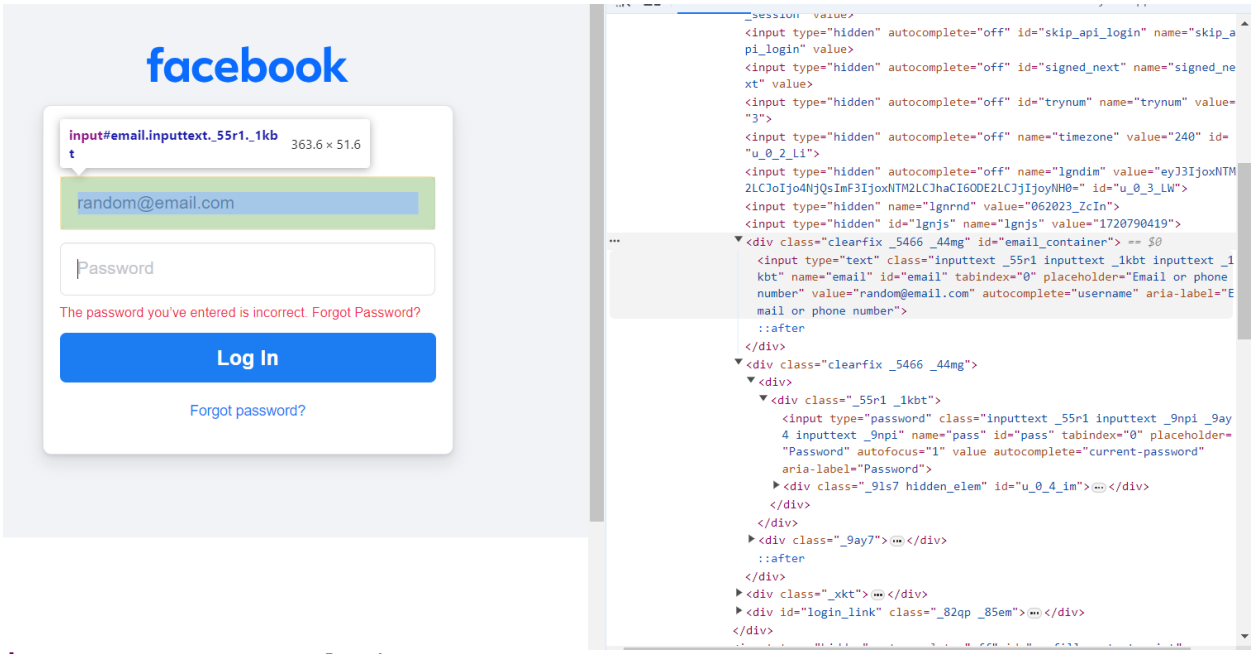
Use the pound sign #, then put the name next to it. An Id can be dynamically changed. Therefore, you have to choose another locator like by name.



14 id have the same name ! avoid it !

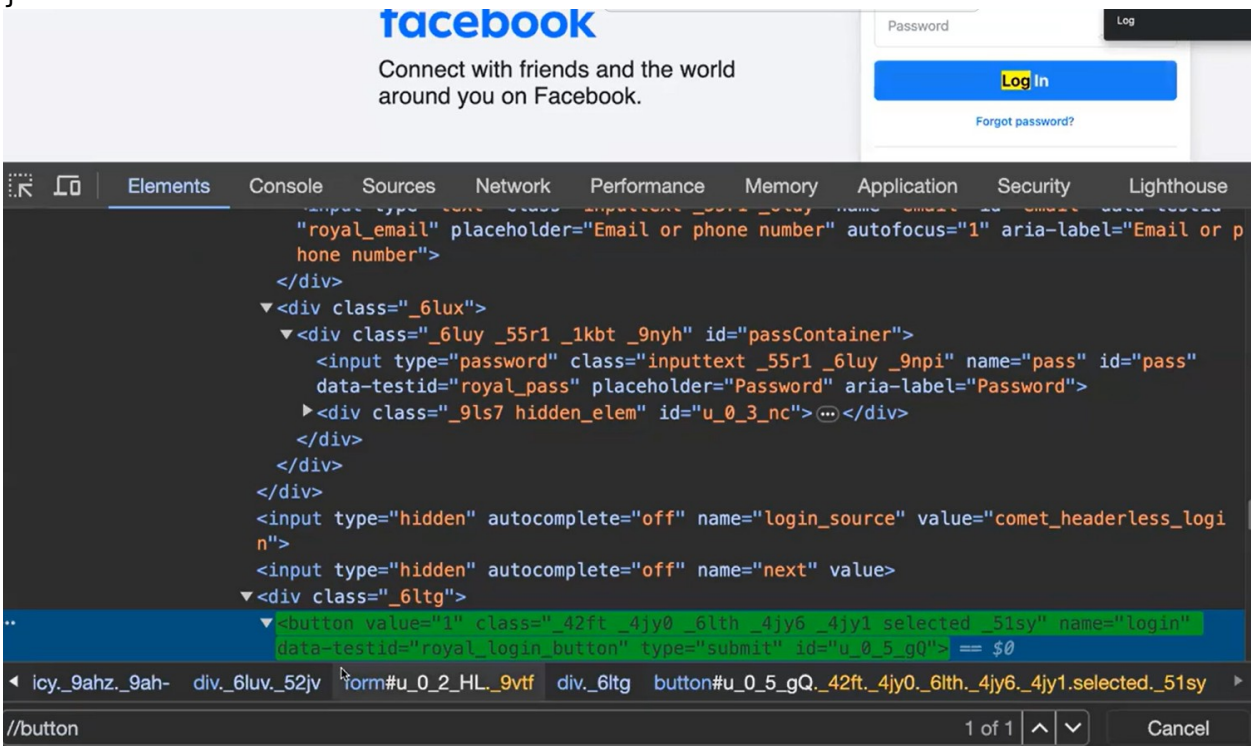


HomeWork：




```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class SecondTest {
    public static void main(String[] args) throws InterruptedException {

        System.setProperty("webdriver.chrome.driver", "resources/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        //First, navigate to actual page
        driver.get("https://www.facebook.com");
//Find the element you want to interact with. (Email textbox)
        driver.findElement(By.id("email")).sendKeys("random@email.com");
//Find the second element (password textbox)
        driver.findElement(By.id("pass")).sendKeys("12345678");
//Click on Login Btn. We are going to use name attr in order to find the btn
//Id attr value changes dynamically, so we are going to use 'name' attribute value,
//which is static. After I found the element, I'm going to use click method
        driver.findElement(By.name("login")).click();
//I'm going to wait for 5 seconds, using Thread.sleep(5000);
        Thread.sleep(5000);
```

```
//Print the title of the new page (page to which we are redirected after clicking login btn)
        System.out.println(driver.getTitle());
        driver.quit();

    }
}
```




```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class LocatorPartTwo {


    public static void main(String[] args) throws InterruptedException {
        // Set the path to the ChromeDriver executable
        System.setProperty("webdriver.chrome.driver", "resources/chromedriver.exe");

        // Create a new instance of the Chrome driver
        WebDriver driver = new ChromeDriver();

        //////Another method://///

//First, navigate to actual page by id and name:
        driver.get("https://www.facebook.com");
//Find the element you want to interact with. (Email textbox)

        WebElement usernameField = driver.findElement(By.id("email"));
        WebElement passwordField = driver.findElement(By.id("pass"));
        WebElement loginBtn = driver.findElement(By.name("login"));
//Find the element using the Tag Name:
       WebElement loginBtnTag = driver.findElement(By.tagName("button"));
```

```
//Sending keys:
        usernameField.sendKeys("random@email.com");
        passwordField.sendKeys("123456");
//After inserting the test, click on login button using click() method
        loginBtn.click();


        // Print the title of the current page
        System.out.println("Page title after login attempt: " + driver.getTitle());
        driver.close();



    }
}
```

Locator by Tag name:

We can choose it **only if it is unique**. Otherwise, need to avoid it.

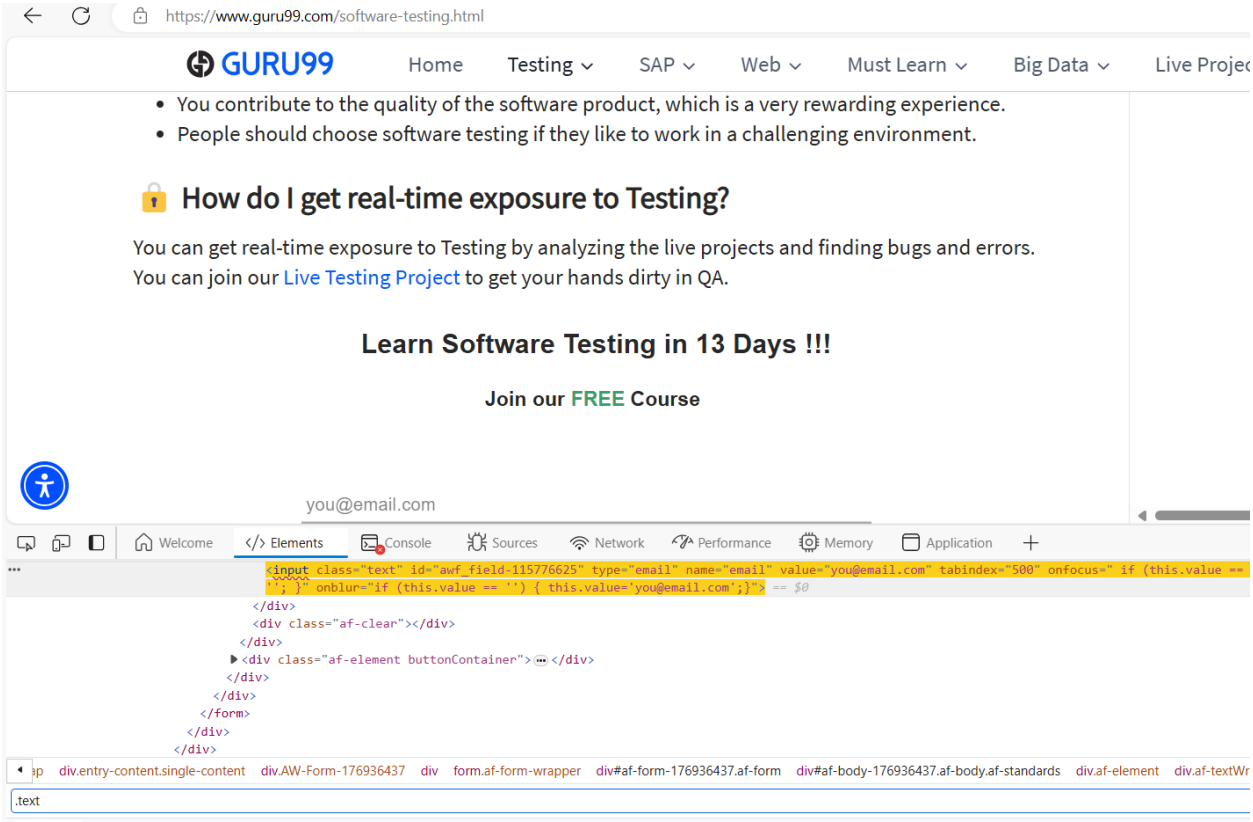
Locator by Class:

Use . dot for **class**, and # for ID when searching.

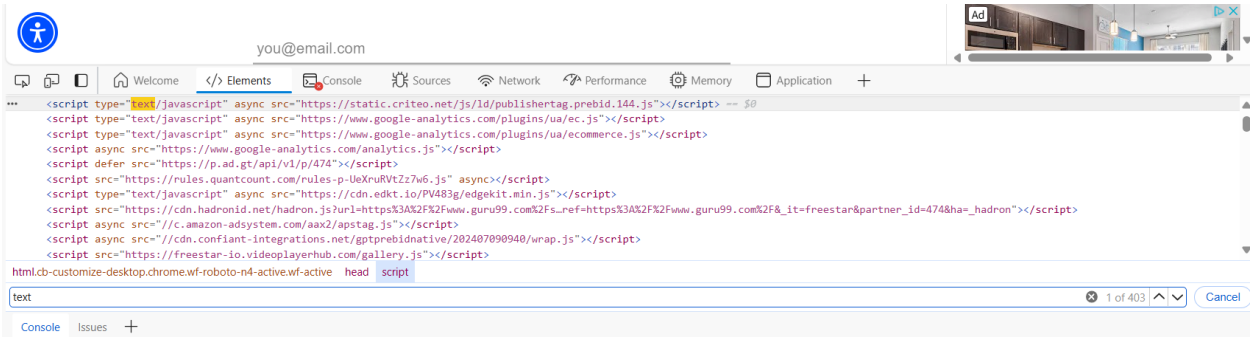
When inspecting the email of guru99:

Software Testing Tutorial (guru99.com)

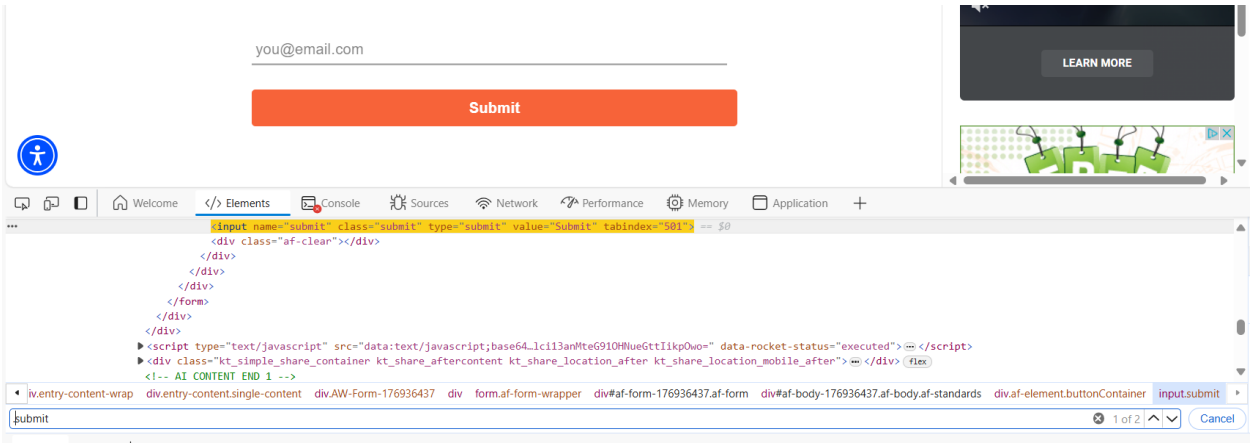
.text has only one element.



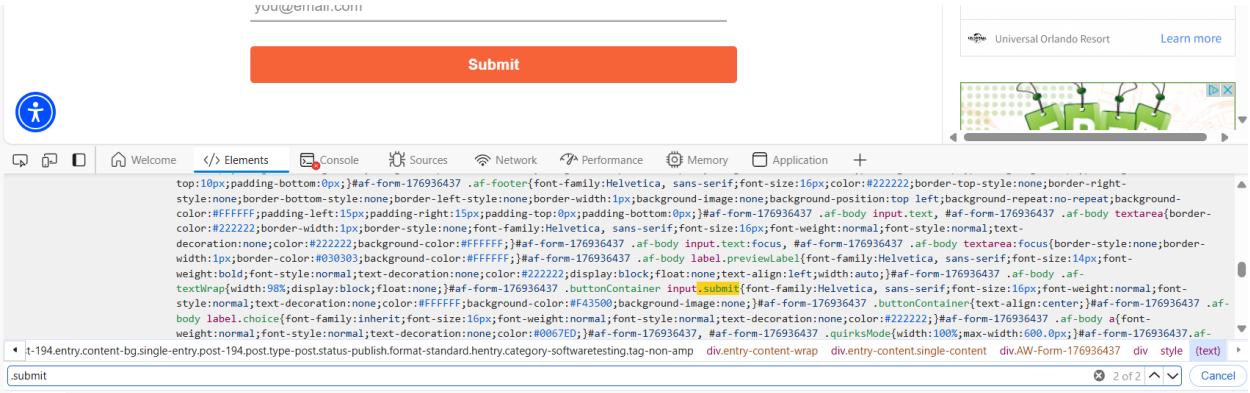
text has 433 elements



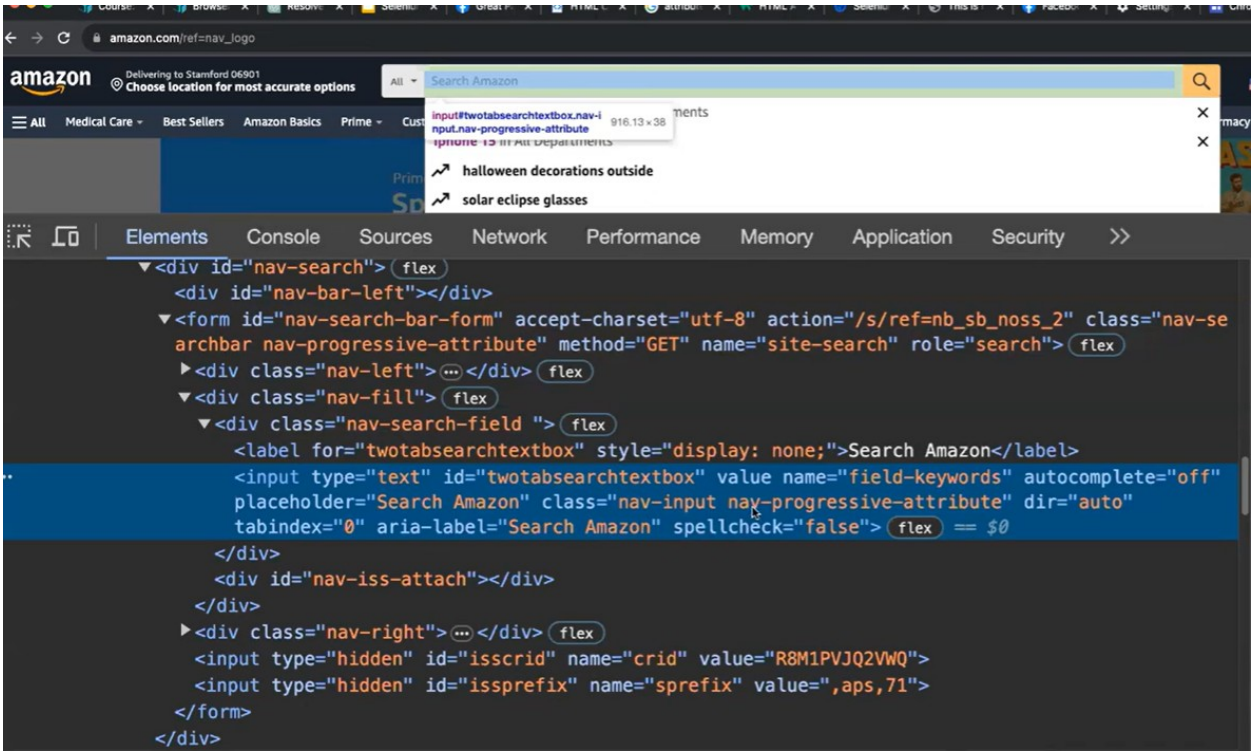
For submit button, it has only one element in the body.



The second one is for the style, it does not cause any problem.



We do not use a class name if it has two separated name with space. it means there two classes responsible for the same element.



StaleElementReferenceException

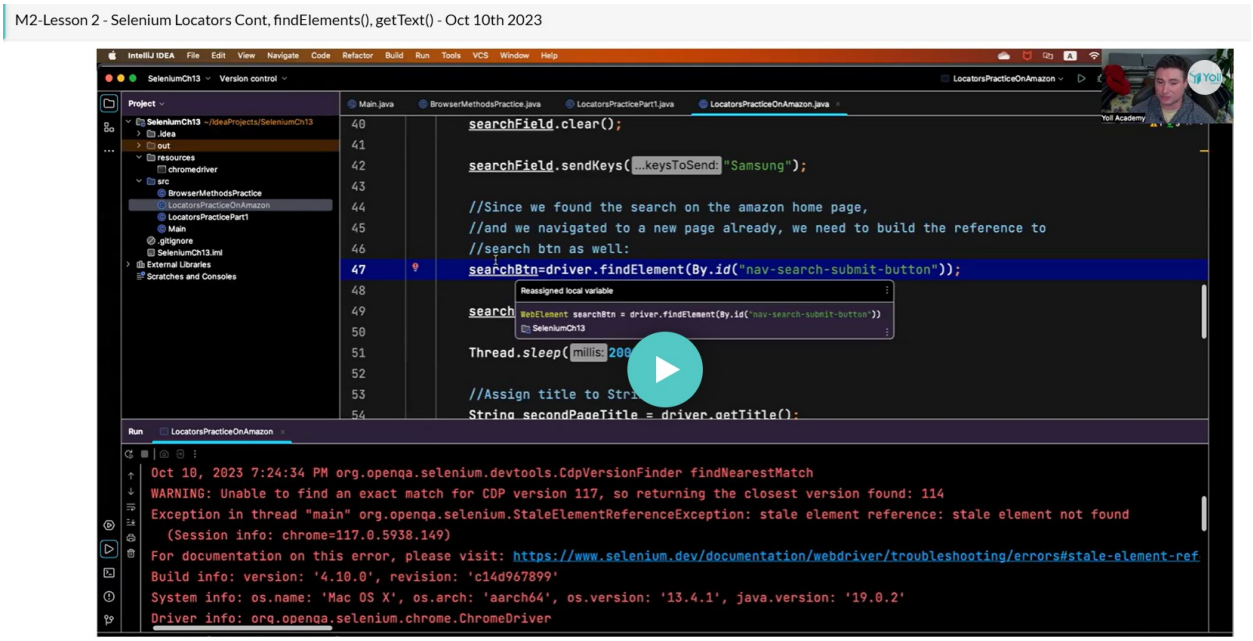
A `StaleElementReferenceException` occurs when an element previously located in the DOM is no longer valid. This can happen due to page reloads, DOM updates, or element removals. Handling this exception involves strategies such as refetching the element, using explicit waits, implementing retry logic, and ensuring the element is present before

performing actions on it. These strategies help to create robust Selenium test scripts that can handle dynamic web pages effectively.

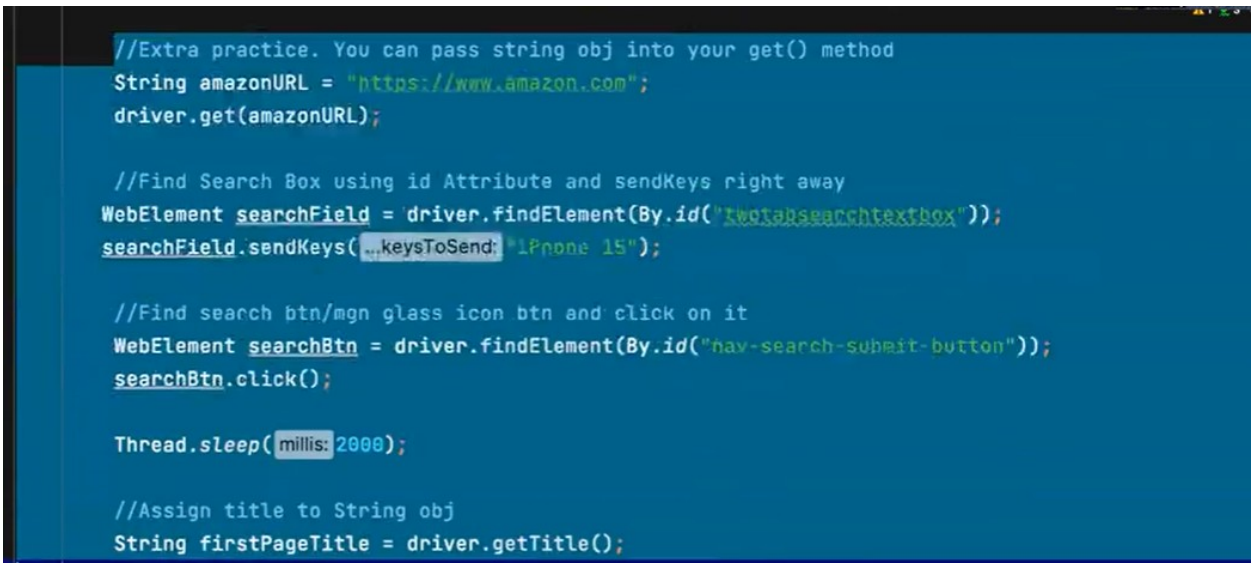
See how to fix this problem on the video:

[M2-Lesson 2 - Selenium Locators Cont, findElements(), getText() - Oct 10th 2023 (yoll.io)]

We need to build the reference again to fix it. It means that we reuse the same code again.


M2-Lesson 2 - Selenium Locators Cont, findElements(), getText() - Oct 10th 2023
searchField.clear();
searchField.sendKeys(...keysToSend: "Samsung");
//Since we found the search on the amazon home page,
//and we navigated to a new page already, we need to build the reference to
//search btn as well:
searchBtn=driver.findElement(By.id("nav-search-submit-button"));
Reassigned local variable
Thread.sleep(millis: 200
//Assign title to Stri
String secondPageTitle = driver.getTitle();
Run
LocatorsPracticeOnAmazon
Oct 10, 2023 7:24:34 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 117, so returning the closest version found: 114
Exception in thread "main" org.openqa.selenium.StaleElementReferenceException: stale element reference: stale element not found
(Session info: chrome=117.0.5938.149)
For documentation on this error, please visit: https://www.selenium.dev/documentation/webdriver/troubleshooting/errors#stale-element-ref
Build info: version: '4.10.0', revision: 'c14d967899'
System info: os.name: 'Mac OS X', os.arch: 'aarch64', os.version: '13.4.1', java.version: '19.0.2'
Driver info: org.openqa.selenium.chrome.ChromeDriver

The problem is caused by the two findelement id:


//Extra practice. You can pass string obj into your get() method
String amazonURL = "https://www.amazon.com";
driver.get(amazonURL);
//Find Search Box using id Attribute and sendKeys right away
WebElement searchField = driver.findElement(By.id("twotabsearchtextbox"));
searchField.sendKeys(...keysToSend: "iPhone 15");
//Find search btn/mgn glass icon btn and click on it
WebElement searchBtn = driver.findElement(By.id("nav-search-submit-button"));
searchBtn.click();
Thread.sleep(millis: 2000);
//Assign title to String obj
String firstPageTitle = driver.getTitle();

We build the two references again:

```
//When we navigate to a new a page or the page got reloaded or updated, we need to build
//the reference once again:
searchField = driver.findElement(By.id("twotabsearchtextbox"));

//Cleer the content of the searchBoxField
searchField.clear();

searchField.sendKeys( ...keysToSend: "Samsung");

//Since we found the search on the amazon home page,
//and we navigated to a new page already, we need to build the reference to
//search btn as well:
searchBtn=driver.findElement(By.id("nav-search-submit-button"));

searchBtn.click();

Thread.sleep( millis: 2000);
```

```
    //Assign title to String obj
    String secondPageTitle = driver.getTitle();

    System.out.println("First page title:"+firstPageTitle+
            "\nSecond page title: "+secondPageTitle);



    driver.quit();
```

By LinkText & partial Link Text

Locates link elements whose visible text matches the given string:

```
<a href="http://www.google.com">Google</a>
```

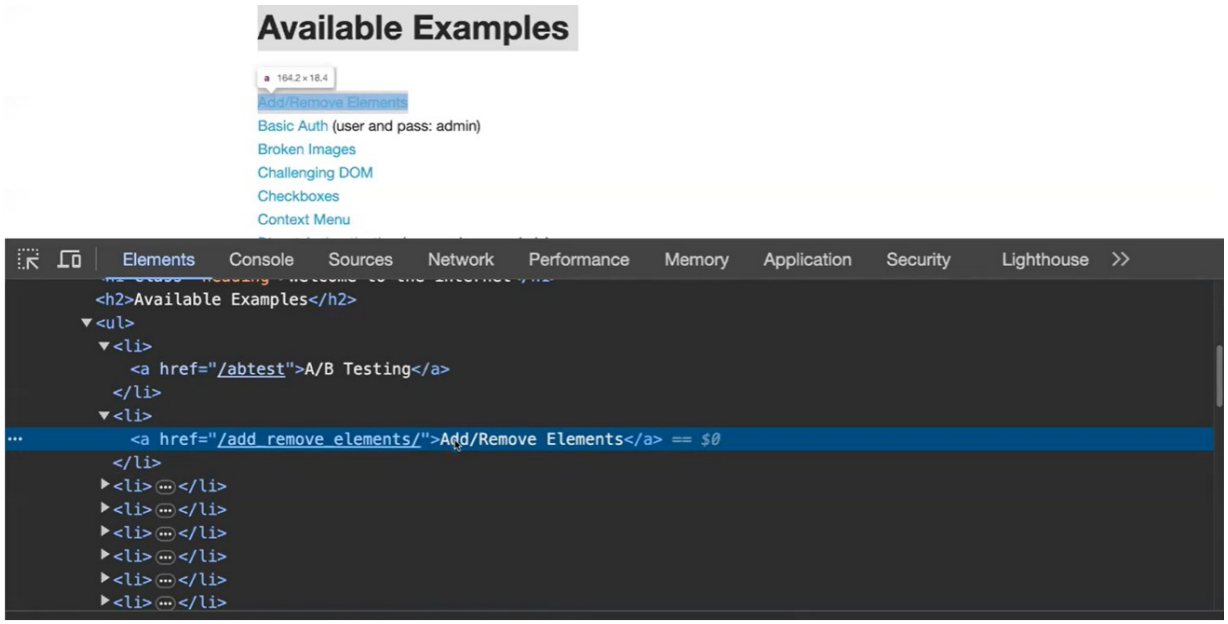
```
driver.findElement(By.linkText("Google"));
```

Locates link elements whose visible text contains the given **Substring**.

```
<ul>
  <li><a href="https://en.wikipedia.org/wiki/Doge_(meme)">
Doge meme</a></li>
  <li>Cat</li>
</ul>
```

```
driver.findElement(By.partialLinkText("Doge "));
```

[The Internet (the-internet.herokuapp.com)](https://the-internet.herokuapp.com)

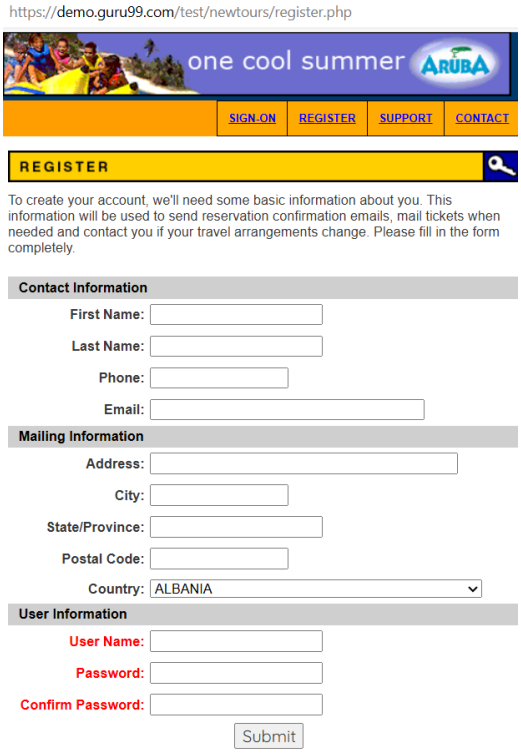



```
  //LinkText and PartialLinkText Practice:
driver.get("https://the-internet.herokuapp.com/");

//find the element using the link text and then click on it
driver.findElement(By.linkText("Add/Remove Elements")).click();
//find the element using the partialLinkText with the first part:
driver.findElement(By.partialLinkText("Add/Remove")).click();
```

Go to http://demo.guru99.com/test/newtours/

1. Click on Register
2. Register an account by filling out all the fields
3. Click on Submit

[Register: Mercury Tours (guru99.com)]




```java
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

import java.util.Random;

public class LocatorPartThree {
    public static void main(String[] args) throws InterruptedException {
        // Set the path to the ChromeDriver executable
        System.setProperty("webdriver.chrome.driver", "resources/chromedriver.exe");

        // Create a new instance of the Chrome driver
        WebDriver driver = new ChromeDriver();
        // Practice finding by class name

        driver.get("https://www.guru99.com/software-testing.html");
```

```
        Thread.sleep(2000);

        // Find the text field for email using name attribute
        WebElement emailField = driver.findElement(By.name("email"));
        // Find the submit button using class name (class name might be incorrect and can
cause NoSuchElementException)
        // WebElement submitBtn = driver.findElement(By.className("submit")); // This
might not work as expected
        emailField.sendKeys("yoll@op.com");
        driver.close();
        // Navigate to "The Internet" website for link text practice
        driver.get("https://the-internet.herokuapp.com/");
        // Find and click the "Add/Remove Elements" link using link text
        driver.findElement(By.linkText("Add/Remove Elements")).click();

        // Find and click the "Add/Remove Elements" link using partial link text
        driver.findElement(By.partialLinkText("Add/Remove")).click();

        // Generate random data for user registration fields
        driver.get("https://the-internet.herokuapp.com/login");
        WebElement usernameField2 = driver.findElement(By.id("username"));
        WebElement passwordField2 = driver.findElement(By.name("password"));
        WebElement confirmPassField = driver.findElement(By.name("confirmPassword"));
        WebElement submitBtn2 = driver.findElement(By.cssSelector(".fa"));

        Random random = new Random();
        int temp = random.nextInt(1000);
        String username = "randomUser" + temp + "@email.com";
        String password = "123456798";

        // Enter user registration data and submit the form
        usernameField2.sendKeys(username);
        passwordField2.sendKeys(password);
        confirmPassField.sendKeys(password);
        submitBtn2.click();

        // Wait for 2 seconds
        Thread.sleep(2000);

        // Close the browser
        driver.quit();

    }
}
```

Find Element vs Find Elements

- **Find Element** → command is used to **identify a (one)** web element within the web page.
  - When using it, it is the responsibility of testers to make sure that web elements are uniquely identifiable using certain properties such as ID, name, className etc.
  - `WebElement loginButton = driver.findElement(By.name("lgn"));`

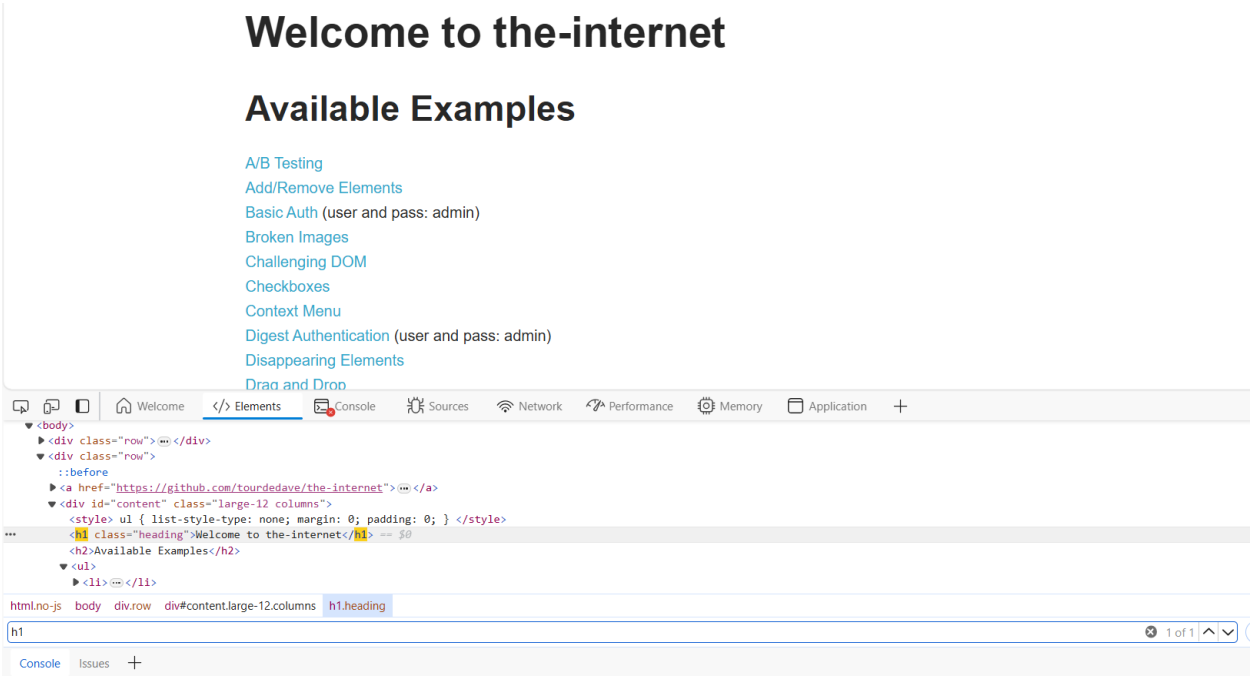
- **Find Elements** → command is used to **identify the list of web elements** within the web page.
  - It means that there are more than one element that matches the given locator.
  - It returns an empty list if there are no elements found using the given locator strategy and locator value.
  - `List<WebElement> pageLinks = driver.findElements(By.name("customLinks"));`

Find Element vs Find Elements (Cont'd)

Find Element	Find Elements
Returns the first most web element if there are multiple web elements found with the same locator	Returns a list of web elements
Throws exception NoSuchElementException if there are no elements matching the locator strategy	Returns an empty list if there are no web elements matching the locator strategy
It will only find one web element	It will find a collection of elements whose match the locator strategy.
Not Applicable	Each Web element is indexed with a number starting from 0 just like an array

```
List<WebElement> elements = driver.findElements(By.name("name"));
 System.out.println("Number of elements:" +elements.size());
 elements.get(0).click();
 elements.get(1).click();
```

Find Element:




```java
import  org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
public class FindElement {
    public static void main(String[] args) throws InterruptedException {
        // Create a new instance of the Chrome driver
        WebDriver driver = new ChromeDriver();
        System.setProperty("webdriver.chrome.driver", "resources/chromedriver");

        //navigate to page the-internet.herokuapp.com
        driver.get("https://the-internet.herokuapp.com/");
        driver.manage().window().maximize();

        //Find the element by using tagName and then extract the text of the element
        // Since we can find two elements that have unique tags on this page,
        // we are going to target these elems
        WebElement headerElem = driver.findElement(By.tagName("h1"));
        WebElement secondaryHeaderElem = driver.findElement(By.tagName("h2"));
        //Extract the text, using getText() method and assign to a string objs
        String headerText = headerElem.getText();
        String subHeaderText = secondaryHeaderElem.getText();
        System.out.println("Header: " + headerText + "\nSubheader: " + subHeaderText);
        driver.close();
    }
}
```

```
Header: Welcome to the-internet
Subheader: Available Examples

Process finished with exit code 0
```

Find Elements:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.List;

public class FindElementsPractice {
	public static void main(String[] args) {
		System.setProperty("webdriver.chrome.driver", "resources/chromedriver.exe");
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();

		// Navigate to page the-internet.herokuapp.com
		driver.get("https://the-internet.herokuapp.com/");


		// TASK: Find all links on the page and then print the text of each link in
the
		// console
		driver.get("https://the-internet.herokuapp.com/");

		// Find all links, using tagName "a"
		List<WebElement> allLinks = driver.findElements(By.tagName("a"));

		// Let's find the size of the List
		int listSize = allLinks.size();
		System.out.println("The size of allLinks ArrayList is: " + listSize);

		// Let's loop through the list and extract the text of each link
		for (WebElement elem : allLinks) {
			System.out.println(elem.getText());
		}
		//another method :regular for loop
		 for(int i=0; i<listSize; i++){
				System.out.println(allLinks.get(i).getText());
	  }

		// Quit the driver
		driver.quit();
	}
}
```

Result:

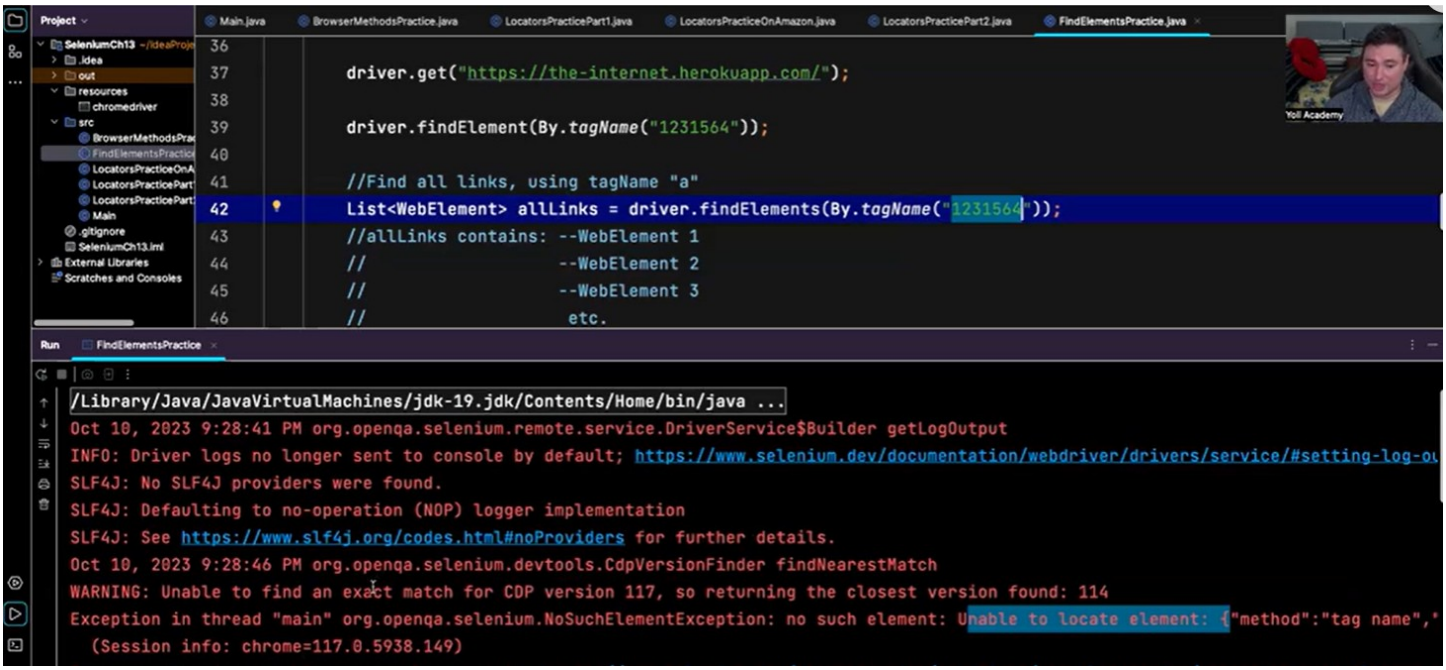
The size of allLinks ArrayList is: 46

A/B Testing
Add/Remove Elements
Basic Auth
Broken Images
Challenging DOM
Checkboxes
Context Menu
Digest Authentication
Disappearing Elements
Drag and Drop

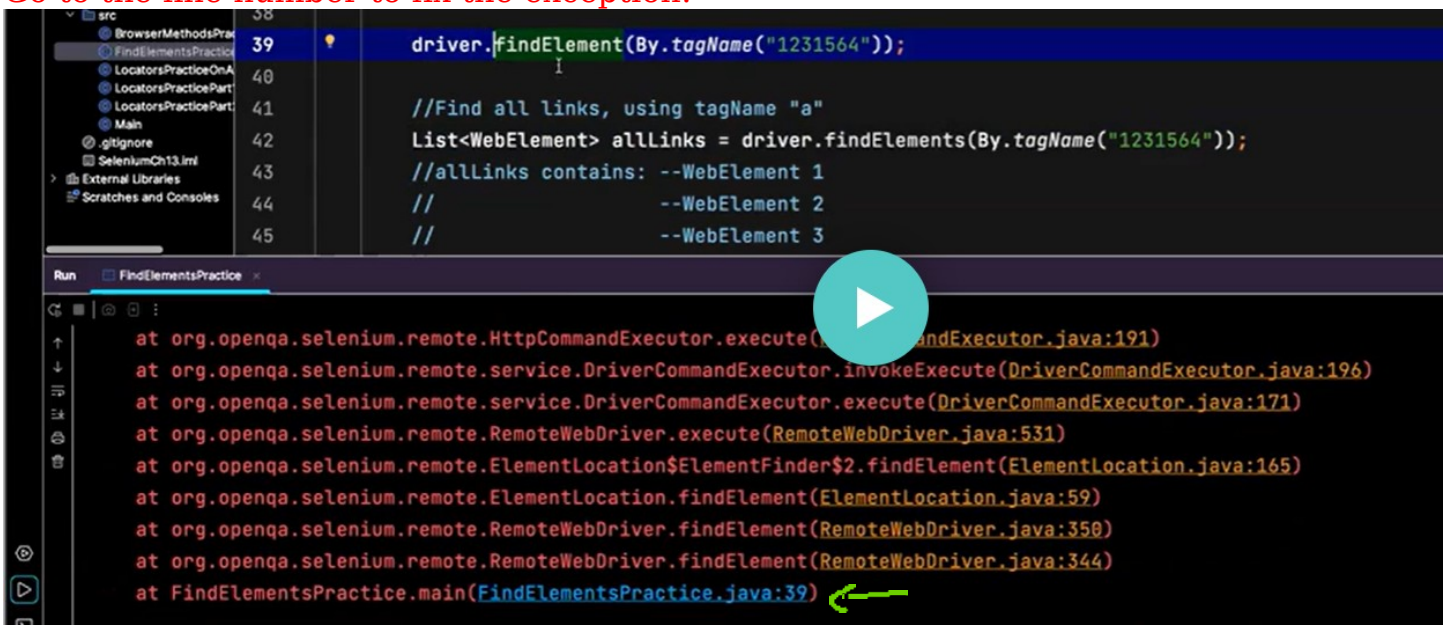
Dropdown
Dynamic Content
Dynamic Controls
Dynamic Loading
Entry Ad
Exit Intent
File Download
File Upload
Floating Menu
Forgot Password
Form Authentication
Frames
Geolocation
Horizontal Slider
Hovers
Infinite Scroll
Inputs
JQuery UI Menus
JavaScript Alerts
JavaScript onload event error
Key Presses
Large & Deep DOM
Multiple Windows
Nested Frames
Notification Messages
Redirect Link
Secure File Download
Shadow DOM
Shifting Content
Slow Resources
Sortable Data Tables
Status Codes
Typos
WYSIWYG Editor
Elemental Selenium

Process finished with exit code 0

If the tag name is not correct or Mispealed, java throws an exception:


Project
Main.java
BrowserMethodsPractice.java
LocatorsPracticePart1.java
LocatorsPracticeOnAmazon.java
LocatorsPracticePart2.java
FindElementsPractice.java
driver.get("https://the-internet.herokuapp.com/");
driver.findElement(By.tagName("1231564"));
//Find all links, using tagName "a"
List<WebElement> allLinks = driver.findElements(By.tagName("1231564"));
//allLinks contains: --WebElement 1
//--WebElement 2
//--WebElement 3
//etc.
Run
FindElementsPractice
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java ...
Oct 10, 2023 9:28:41 PM org.openqa.selenium.remote.service.DriverService$Builder getLogOutput
INFO: Driver logs no longer sent to console by default; https://www.selenium.dev/documentation/webdriver/drivers/service/#setting-log-ou
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See https://www.slf4j.org/codes.html#noProviders for further details.
Oct 10, 2023 9:28:46 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 117, so returning the closest version found: 114
Exception in thread "main" org.openqa.selenium.NoSuchElementException: no such element: Unable to locate element: {"method":"tag name",*
(Session info: chrome=117.0.5938.149)

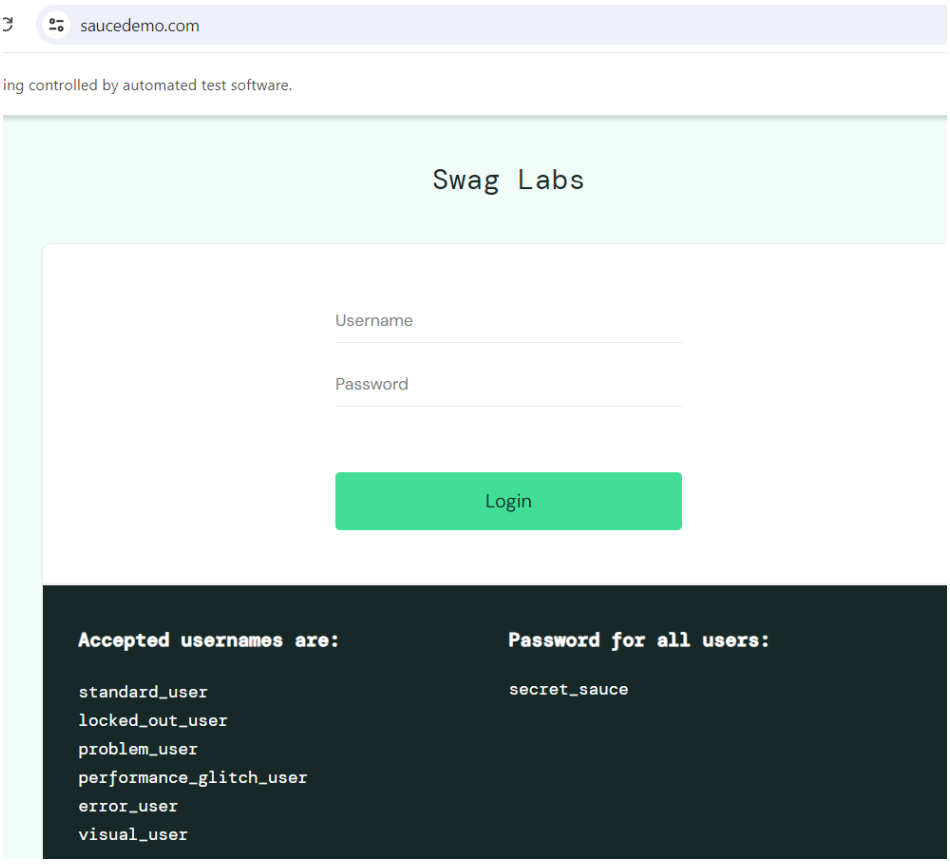
Go to the line number to fix the exception:

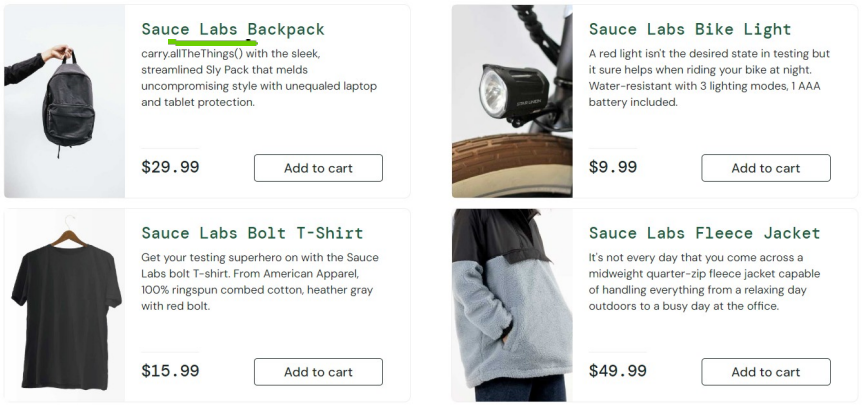


Homework : Locators

```
/*
```

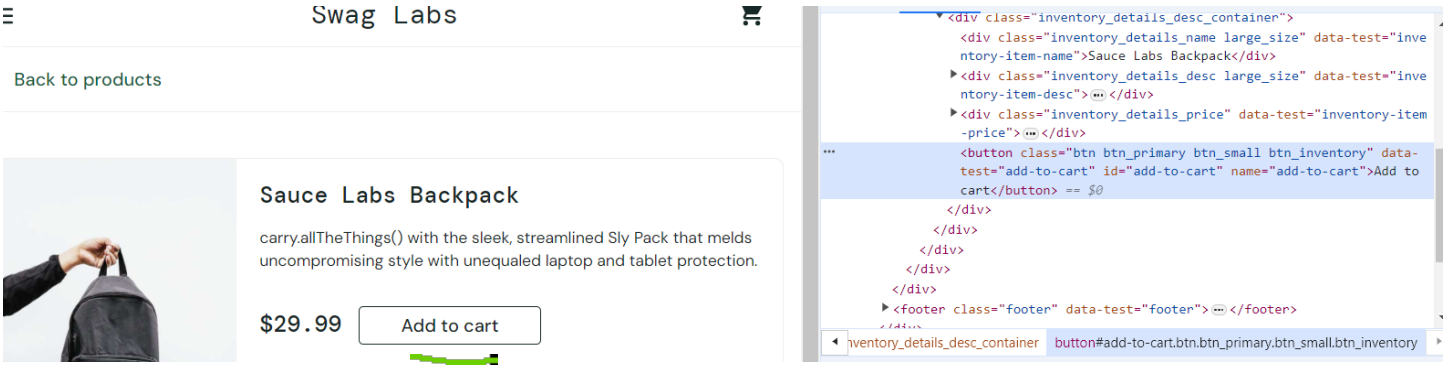
```
Navigate to "https://www.saucedeno.com
Find following elements:
Text Boxes: Username and Password using ID attribute
Button: Login
using name attribute
Use following values to send text
Username: standard_user
Password: secret_sauce
If you logged in successfully, click on 'Sauce Labs Backpack' link using any locator you
like Add Thread.sleep(2000);
Click on Add to Cart button using any locator you like
Add Thread.sleep(2000);
Click on Shopping cart logo (in upper right corner) using class attribute
Add Thread.sleep(5000);
Close/quit driver.
*/
```



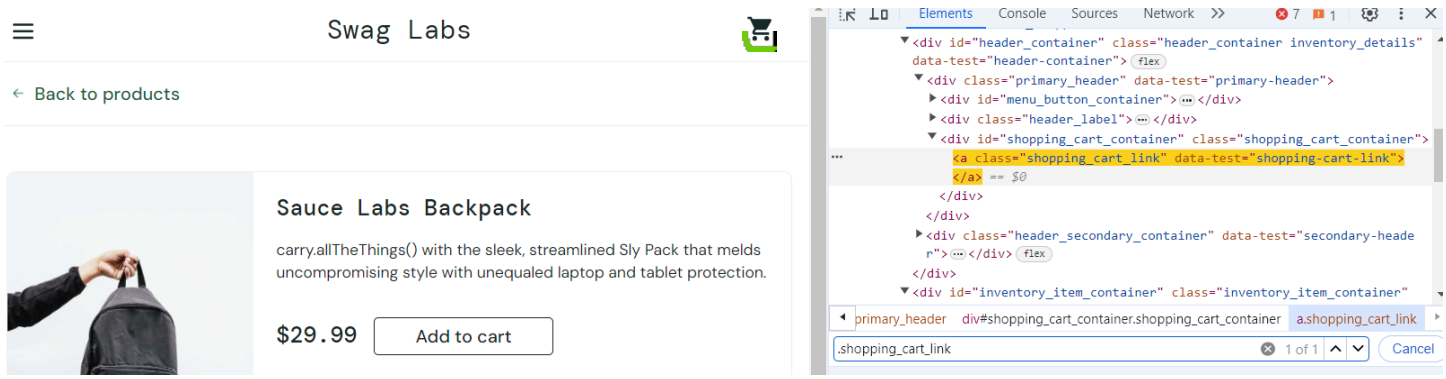

Sauce Labs Backpack
carry.allTheThings() with the sleek, streamlined Sly Pack that melds uncompromising style with unequaled laptop and tablet protection.
$29.99
Add to cart
Sauce Labs Bike Light
A red light isn't the desired state in testing but it sure helps when riding your bike at night. Water-resistant with 3 lighting modes, 1 AAA battery included.
$9.99
Add to cart
Sauce Labs Bolt T-Shirt
Get your testing superhero on with the Sauce Labs bolt T-shirt. From American Apparel, 100% ringspun combed cotton, heather gray with red bolt.
$15.99
Add to cart
Sauce Labs Fleece Jacket
It's not every day that you come across a midweight quarter-zip fleece jacket capable of handling everything from a relaxing day outdoors to a busy day at the office.
$49.99
Add to cart

Hint :Selenium is smart enough to recognize that div tag corresponds to a tag (link)

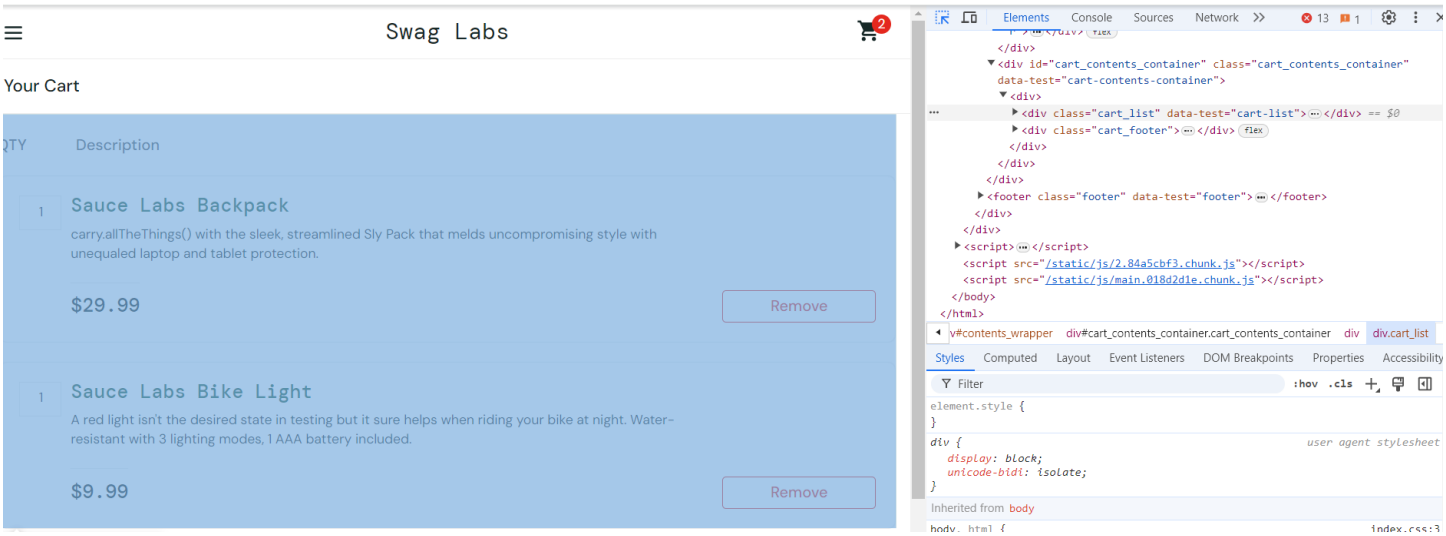
```
WebElement addToCartButton = driver.findElement(By.id("add-to-cart"));
```



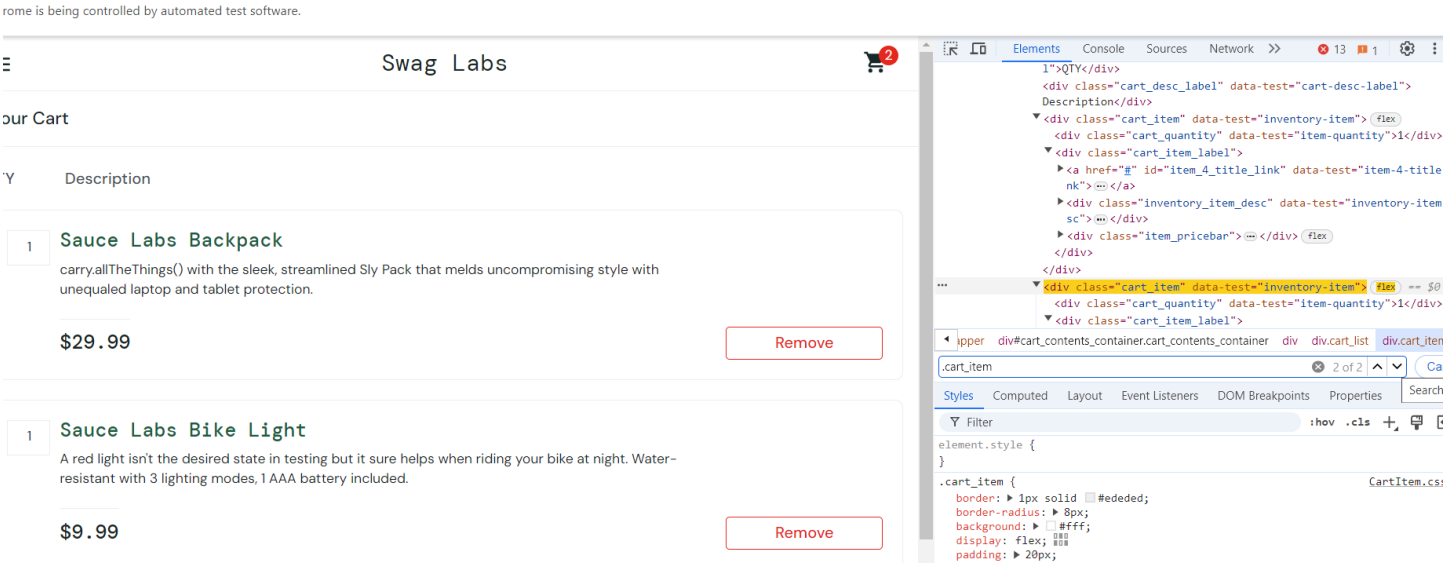

```
WebElement shoppingCartButton = driver.findElement(By.className("shopping_cart_link"));
```



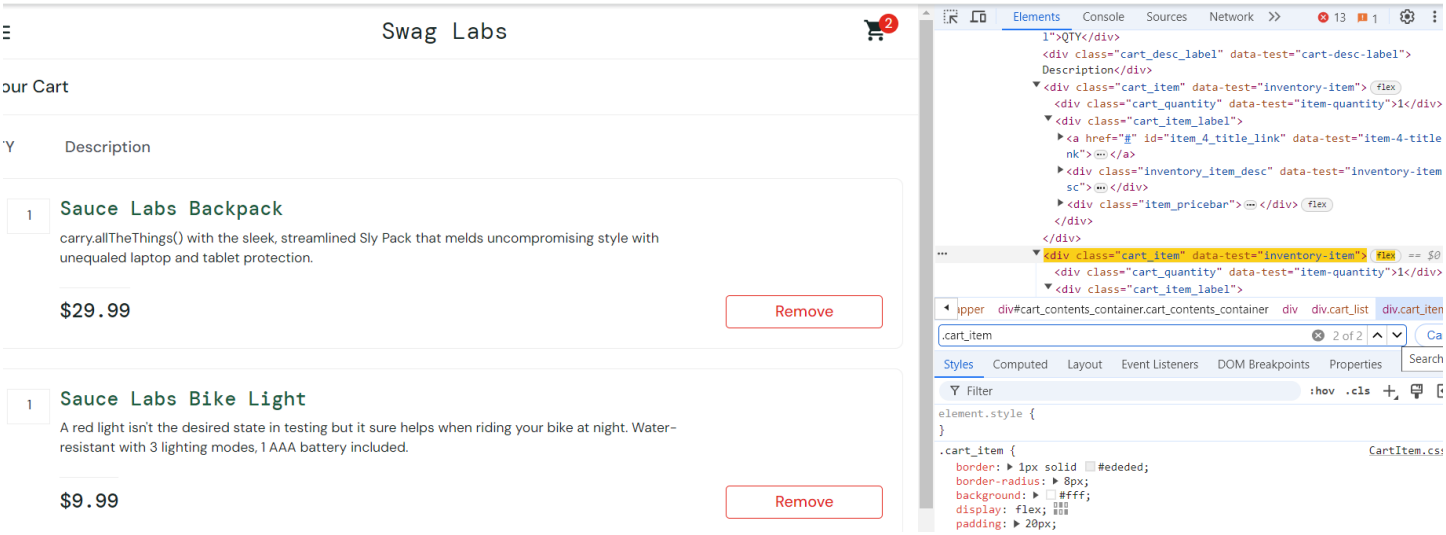

```
List<WebElement> itemsInCart = driver.findElements (By.className("cart_item"));
```



.cart_item : we have two element for the two checked items.



rome is being controlled by automated test software.




```java
package module2;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

import java.util.List;

public class PracticeSauceLabs {
	public static void main(String[] args) throws InterruptedException {


		WebDriver driver = new ChromeDriver();
// This line of code maximizes the window
		driver.manage().window().maximize();
// Navigate to "https://www.saucedemo.com
		driver.get("https://www.saucedemo.com");
// Make the execution of code pause for 2 seconds
		Thread.sleep(2000);// Find following elements:
		// Text Boxes: Username and Password using ID attribute
		// Button: Login using name attribute
		WebElement usernameField = driver.findElement(By.id("user-name"));
		WebElement passwordField = driver.findElement(By.id("password"));
		WebElement loginButton = driver.findElement(By.name("login-button"));
// Having the elements located fill out the username password and click login button
		usernameField.sendKeys("standard_user");
		passwordField.sendKeys("secret_sauce");
		loginButton.click();
// Make the execution of code pause for 2 seconds
		Thread.sleep(2000);
// We want to locate the backpack item title and click on it
		WebElement backpackItemTitle = driver.findElement(By.linkText("Sauce Labs
Backpack"));
		backpackItemTitle.click();
// Make the execution of code pause for 2 seconds
		Thread.sleep(2000);
		WebElement addToCartButton = driver.findElement(By.id("add-to-cart"));
		addToCartButton.click();
```

```
// Make the execution of code pause for 2 seconds
            Thread.sleep(2000);
            WebElement shoppingCartButton =
driver.findElement(By.className("shopping_cart_link"));
            shoppingCartButton.click();
// Make the execution of code pause for 5 seconds
            Thread.sleep(5000);
// Validate that there is one item in cart
            // - I want to identify and store all the web elements in cart
            // - I already have a locator that does it :class cart_item
            List<WebElement> itemsInCart =
driver.findElements(By.className("cart_item"));
            System.out.println("In the cart are: " + itemsInCart.size() + " items.");
            if (itemsInCart.size() == 1) {
                  System.out.println("TEST: PASSED");
            } else {
                  System.err.println("TEST: FAILED");

            }
            driver.quit();
      }
}
```

```
"C:\Program Files\Java\jdk-22\bin\java.exe" ...
In the cart are: 1 items.
TEST: PASSED

Process finished with exit code 0
```

```
if (itemsInCart.size() == 5) {
                  System.out.println("TEST: PASSED");
            } else {
                  System.err.println("TEST: FAILED");

            }
```

```
TEST: FAILED
In the cart are: 1 items.

Process finished with exit code 0
```

XPATH Locator:

Relative Xpath

Basic Xpath

We can use Xpath to write expressions to locate elements, based on other attributes:

- //input[@name='uid'] → Uses name as attribute to locate.
- //a[@id='search'] → Uses id
- //select[@class='offerings'] → Uses class
- //img → Uses just tag name
- //div[@value='RESET'] → Uses value attribute
- //a[@href='http://demo.com/'] → uses link
- //img[@src='//cdn.amazon.com/images/cake.png'] → uses src attribute
- //*[@name='fname'] → * means any tagname.
- //*[@name='fname']/input → / single slash means the child element of the specified element with input tag.

/.. → means going to parent element

Basic Web Element - Button

```
<button type="submit" class="btn btn_primary" title="subscribe button" area-
lable="subscribe.button
">Subscribe</button>
// Base XPath
//button [@title = 'subscribe button']

// Identifying all img tags
<body>
<div>
<img src="path/to/image1.jpg" alt="Description of Image 1" />
<img src="path/to/image2.jpg" alt="Description of Image 2" />
</div>
</body>
//img -> 2 -> driver.findElements() -> List<WebElemet>


Moving to next generation
Use: /
<div id="parentDiv1">
<span class="childSpan">Hello, World!</span>
<span name="seleniumSpan">Selenium Class!</span> </div>
<div id="parentDiv2">
<span class="childSpan">Hello, Students!</span> </div>
//div[@id = 'parentDiv1']/span[@class = 'childSpan']
```

Moving through multiple generations :
Use // to skip muliple children

```
<div id="parentDiv]
<span class="childSpan">Hello, World!</span> <span name="seleniumSpan">
Selenium Class!
<div class="firstGen">
First Generation
<div class="secondGen"> Second Generation <div class="thirdGen"> Third Generation
<p class="finalGen">Final Generation</p>
</div>
</div>
</div>
</span>
</div>
<div id="parentDiv2">
<span class="childSpan">Hello, Students!</span> </div>
// first approach is combining // with /
//div[@id = 'parentDiv1']/span [@name='seleniumSpan']/div/div/div
//div[@id='parentDiv1']//div[@class='thirdGen']
//div[@id='parentDiv1']//p
```

Moving back to parent tag
Use : /..

```
<div class="new_class">
<p id="childPara1">Welcome!</p> </div>
<div class="new_class">
<p id="childPara2">Bye!</p> </div>

//p[@id= 'childPara1']/..
```

Using indicies in xpath
The index starts with the number ONE NOT ZERO

```
<div class="container1">
<p class="info">Data 1</p>
<p class="info">Data 2</p>
<p class="info">Data 3</p>
<p class="info">Data 4</p> </div>
<div class="container2">
<p class="info">Data 1</p>
<p class="info">Data 2</p>
<p class="info">Data 3</p>
<p class="info">Data 4</p>
</div>

//div[@class = 'container1']/p[3]
(//p[@class = 'info']) [4]
//p[@class = 'info'] -> List
//*[@class ='btn-unstyled']
```

→ returns multiple webElements

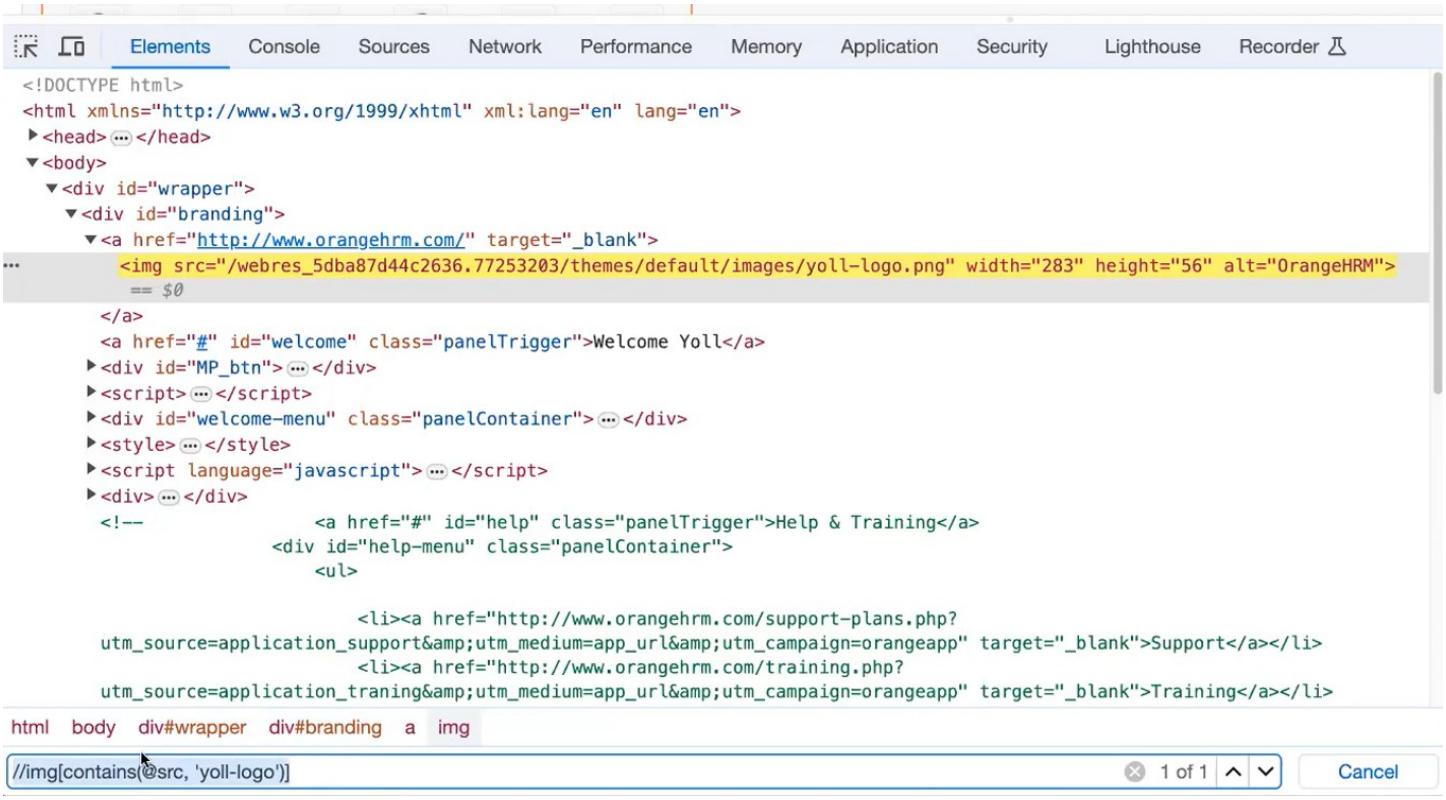
●If we want to specify the element we want to click we can use index

```
(//*[@class ='btn-unstyled'])[1]
```

(//*[@class ='btn-unstyled'])[2]
● Index is not reliable, therefore should be used when there is no other good choice.

Xpath for image using contains: //img[contains(@src,’yoll-logo’)]

CSS : img[src*=’yoll-logo’]



XPath for Dynamic Elements

- **Text()** → with text function, we find the element with **exact text match**
  - `//*[text() ='whatever']`

- **Contains** → is a method used in XPath expression. It has an ability to find the element with partial text:
  - `//*[contains(@name,'btn')]` → we have taken the 'name' as an attribute and 'btn' as an partial value
  - `//input[contains(@placeholder, 'sername')]`
- **Partial text**
  - `//*[contains(text(),'what')]`
- **Using OR & AND** → We can combine xpath expression with logical operator OR & AND
  - `//*[@type='submit' or @name='btnReset']`
  - `//input[@type='submit' and @name='btnLogin']`

- **Starts-with** → match the starting text of the attribute, in cases where rest of the element value changes.
  - E.g. In a page there could be 3 elements with IDs like: message34, message255, message123
  - `//label[starts-with(@id,'message')]` → This expression will identify all three elements

```
// In xpath we can use functions/methods
// text() - text is used instead of attribute
//tag [@attribute = 'value']
//tag [text() = 'value']
// contains() function
// it is directly used in xpath []
// contains(@attribute, 'partial value')
// contains(text(), 'partial value')
```

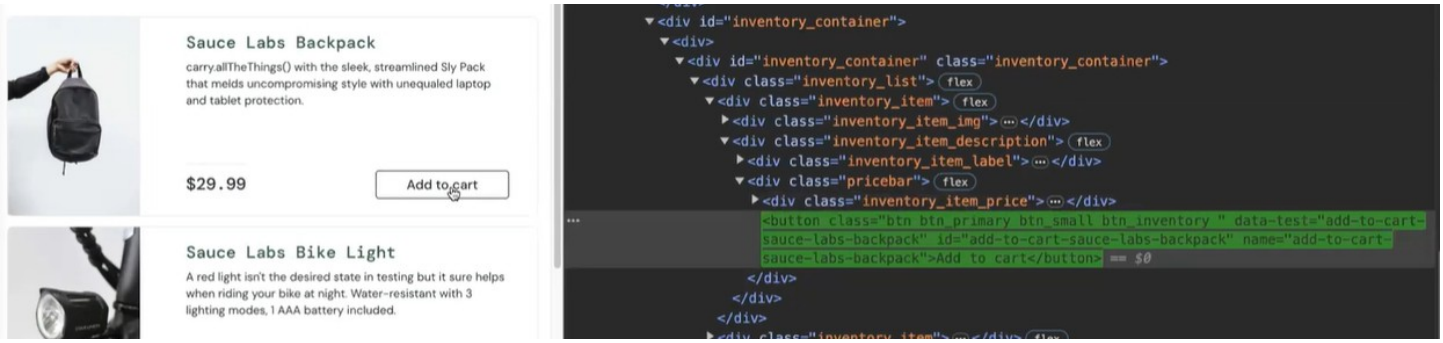
Use different functions to locate elements

```
<div class="content">
<p>Welcome to the world of XPath!</p>
<p>This is a tutorial on using the text() function.</p> <ul>
<li>Item 1: Introduction</li>
<li>Item 2: Basics</li>
<li>Item 3: Advanced</li>
</ul>
<p>Let's dive deeper into XPath functions.</p>
</div>
//li[text() = 'Item 3: Advanced']
//li[contains(text(), 'Item 3')]
```

Using Logical Operators in xpath // 'and' operator

```
<body>
<div>
<input name="sampleInput" class="inputClass" type="username">
<input name="sampleInput" class="inputClass" type="password">
<input name="sampleInput" class="differentClass" type="text">
</div> <div>
<input name="anotherInput" class="inputClass" type="username">
<input name="anotherInput" class="inputClass" type="password">
<input name="anotherInput" class="differentClass" type="text">
```

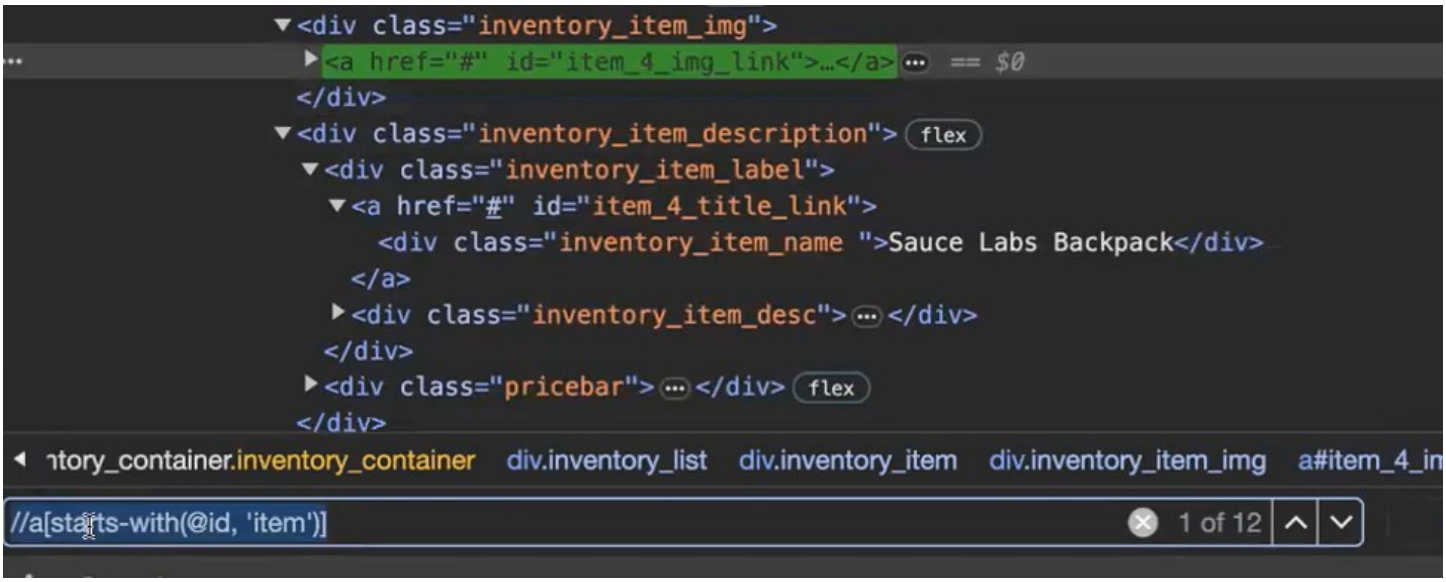
```
</div> </body>
//input [@name='sampleInput']-> 3 elements
//input [@name='sampleInput' and @type='text' and @class = 'differentClass']
//input [@type = 'username' or @type = 'password' and @name = 'sampleInput']
// We can use logical operators to filter for a specific element
// We can use as operands the attrinbutes or functions as text() contains etc...

If the id value is dynamic and changing to another value, we can use OR :
Add to cart Button has: id='add-to-cart-sauce-labs-backpack
Then it canged to Remove, the id is: id = 'remove-sauce-labs-backpack'
The XPATH will be:

//button[@id='add-to-cart-sauce-labs-backpack' or @id = 'remove-sauce-labs-backpack']
```



Remove Button:

Starts with:

```
<div class="content">
<p>Welcome to the world of XPath!</p>
<p>This is a tutorial on using the text() function.</p> <ul>
<li>Item 1: Introduction</li>
<li>Item 2: Basics</li>
<li>Item 3: Advanced</li>
</ul>
<p>Let's dive deeper into XPath functions.</p>
</div>

//li[starts-with(text(), 'Item 2')]
//div[starts-with(@class, 'con')]

Example starts with item for a class.
```



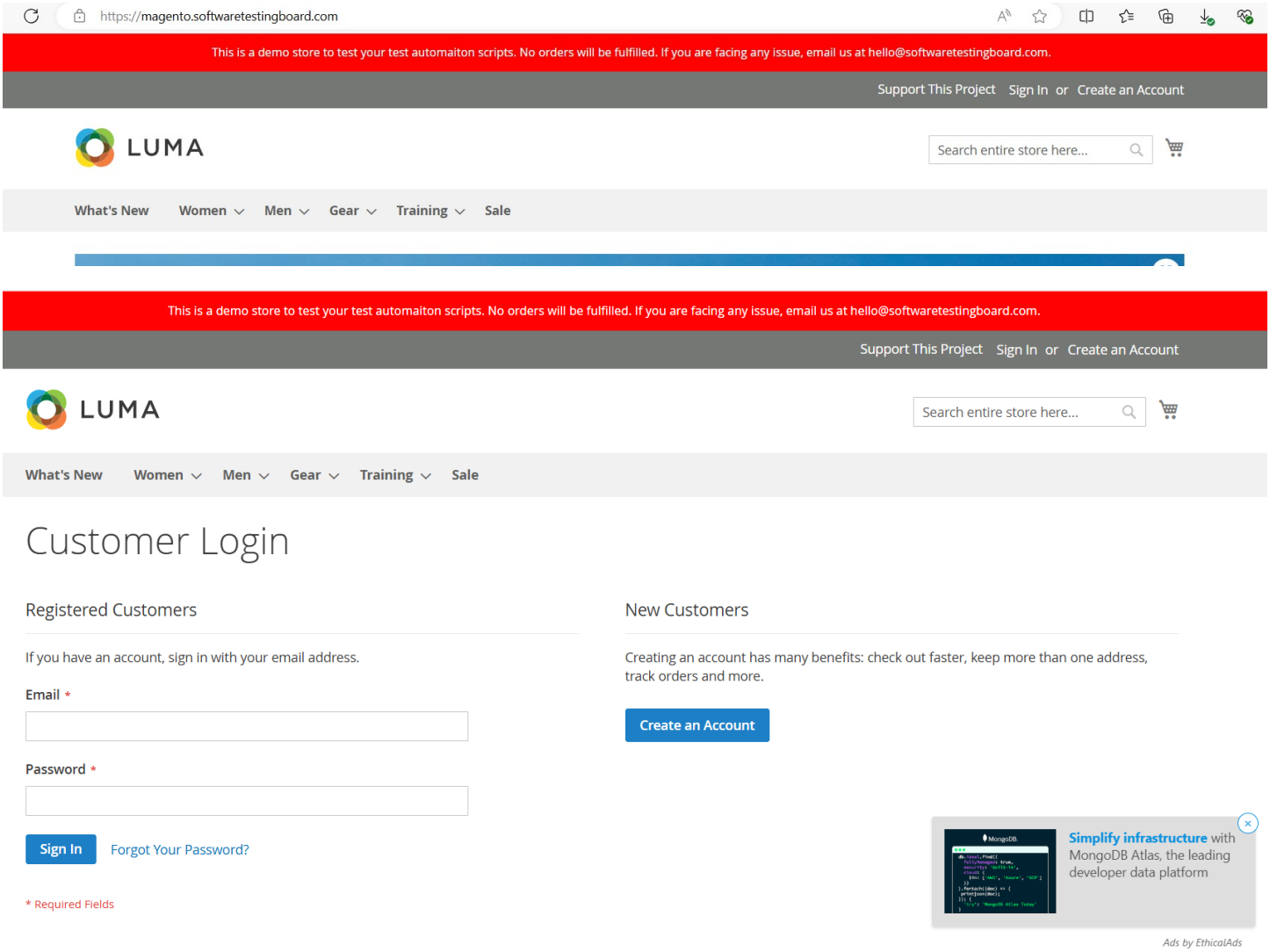
Task ???????? to be done

https://magento.softwaretestingboard.com/

Locate below elements with xpath

- Logo
- Cart
- Search input box
- “Contact Us” link
- “Subscribe” link

- Go to http://automationpractice.com/index.php(alternative: https://magento.softwaretestingboard.com/)
- Complete below test case
  - Setup your email and pwd by creating string variables for each on your IDE
    - UN: ystudent@email.com (Magento Store: yollstudent@email.com)
    - PW: secret (Magento Store: Pa$$vv0rd)
  - Click signin button on homepage using xpath and ‘contains’ method (for Magento use index)
  - Enter your email using xpath and ‘id’ attribute
  - Enter your pwd using xpath and ‘name’ attribute
  - Click signin button using ‘id’ only (Magento Store: id and class attributes)
  - Print Title of the page to verify you successfully signed in (Automation Practice Only)
  - Print Welcome Message (Magento Store Only: Use contains and text for xpath and getText())
  - Close the browser


https://magento.softwaretestingboard.com
This is a demo store to test your test automaiton scripts. No orders will be fulfilled. If you are facing any issue, email us at hello@softwaretestingboard.com.
Support This Project Sign In or Create an Account
LUMA
Search entire store here...
What's New Women Men Gear Training Sale
This is a demo store to test your test automaiton scripts. No orders will be fulfilled. If you are facing any issue, email us at hello@softwaretestingboard.com.
Support This Project Sign In or Create an Account
LUMA
Search entire store here...
What's New Women Men Gear Training Sale
Customer Login
Registered Customers
If you have an account, sign in with your email address.
Email *
Password *
Sign In
Forgot Your Password?
* Required Fields
New Customers
Creating an account has many benefits: check out faster, keep more than one address, track orders and more.
Create an Account
Simplify infrastructure with MongoDB Atlas, the leading developer data platform
Ads by EthicalAds

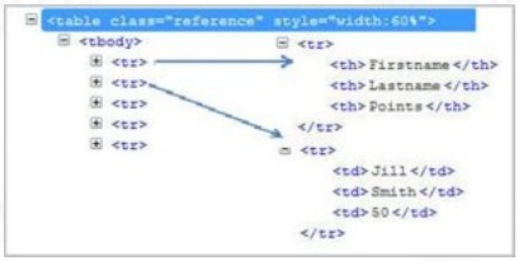
```java
package module2;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class MagentoPracticeTask {
    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver", "resources/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();

        // Store the credentials to sign in
        String username = "yollstudent@email.com";
        String password = "Pa$$vv0rd";

        // Navigate to the website
        driver.get("https://magento.softwaretestingboard.com/");
```

```
        // Wait for 2 seconds for the page to render
        Thread.sleep(2000);


        //store the Sign In button
        //USE SINGLE QUOTES NOT DOUBLE QUOTES LIKE THIS
        //WebElement signInButton = driver.findElement(By.xpath(  "//header[@class
=\"page-header\"]//a[contains(text(), \"sign_In\")]"));
        //USE SINGLE QUOTES, AND USE THE PARENT BECAUSE THERE ARE TWO SIGN IN BUTTON
        WebElement signInButton = driver.findElement(By.xpath("//header[@class='page-
header']//a[contains(text(), 'Sign In')]"));
        signInButton.click();

        // Wait for 2 seconds for the page to render
        Thread.sleep(2000);

        WebElement usernameField = driver.findElement(By.xpath("//input[@id='email']"));
        WebElement passwordField = driver.findElement(By.xpath("//input[contains(@title,
'Password')]"));
        WebElement logInButton = driver.findElement(By.xpath("//fieldset//button"));

        usernameField.sendKeys(username);
        passwordField.sendKeys(password);
        logInButton.click();

        // Wait for a few seconds to observe the result
        Thread.sleep(2000);

        // Quit the driver
        driver.quit();
    }
}
```

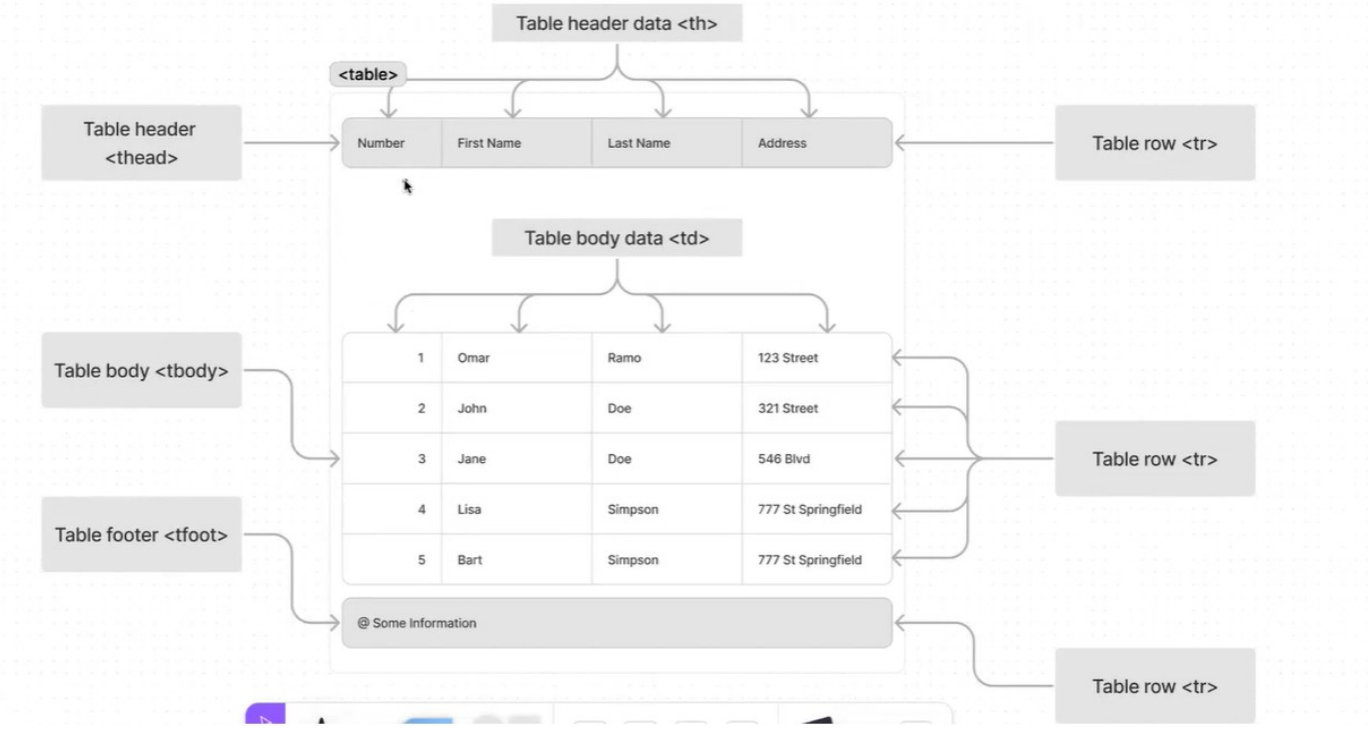
WEBTABLES

- Web tables are basically a group of elements that are logically stored in a row and column format. It is used to organize similar information on a web page.

Firstname	Lastname	Points
Jill	Smith	50
Eve	Jackson	94
John	Doe	80
Adam	Johnson	67



- **'table'** tag → defines HTML table.
- **'thead'** tag → defines container for table header.
- **'tbody'** tag → defines a container for rows and columns in table body.
- **'tr'** → defines rows in an HTML table.
- **'td'/'th'** → define the column of an HTML table. (th = table header)



Omar: tbody/tr[1]/td[2]

John: tbody/tr[2]/td[2]

Jane: tbody/tr[3]/td[2]

Lisa: tbody/tr[4]/td[2]

Bart: tbody/tr[5]/td[2]

Ramo: tbody/tr[1]/td[3]

Doe: tbody/tr[2]/td[3]

Doe: tbody/tr[3]/td[3]

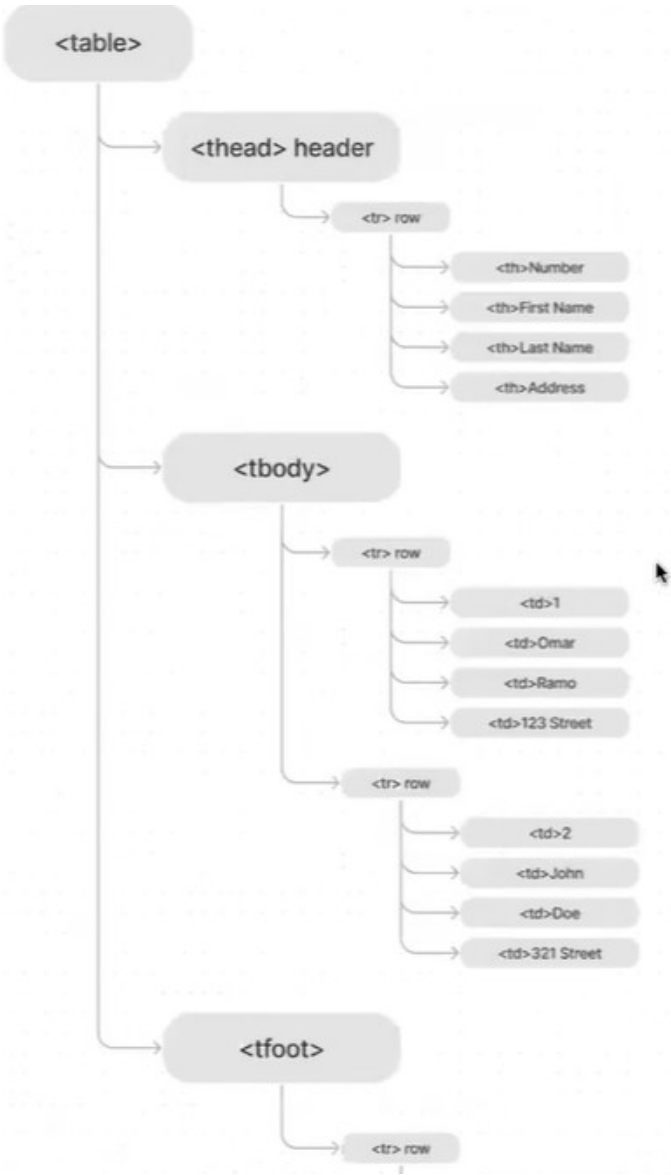
Simpson: tbody/tr[4]/td[3]

Simpson: tbody/tr[5]/td[3]

Omar – John – Jane- Lisa- Bart : tbody/tr/td[2]

2- John- Doe-321 Street : tbody/tr[2]/td

//table[@id='productTable']/tbody/tr[2]/td

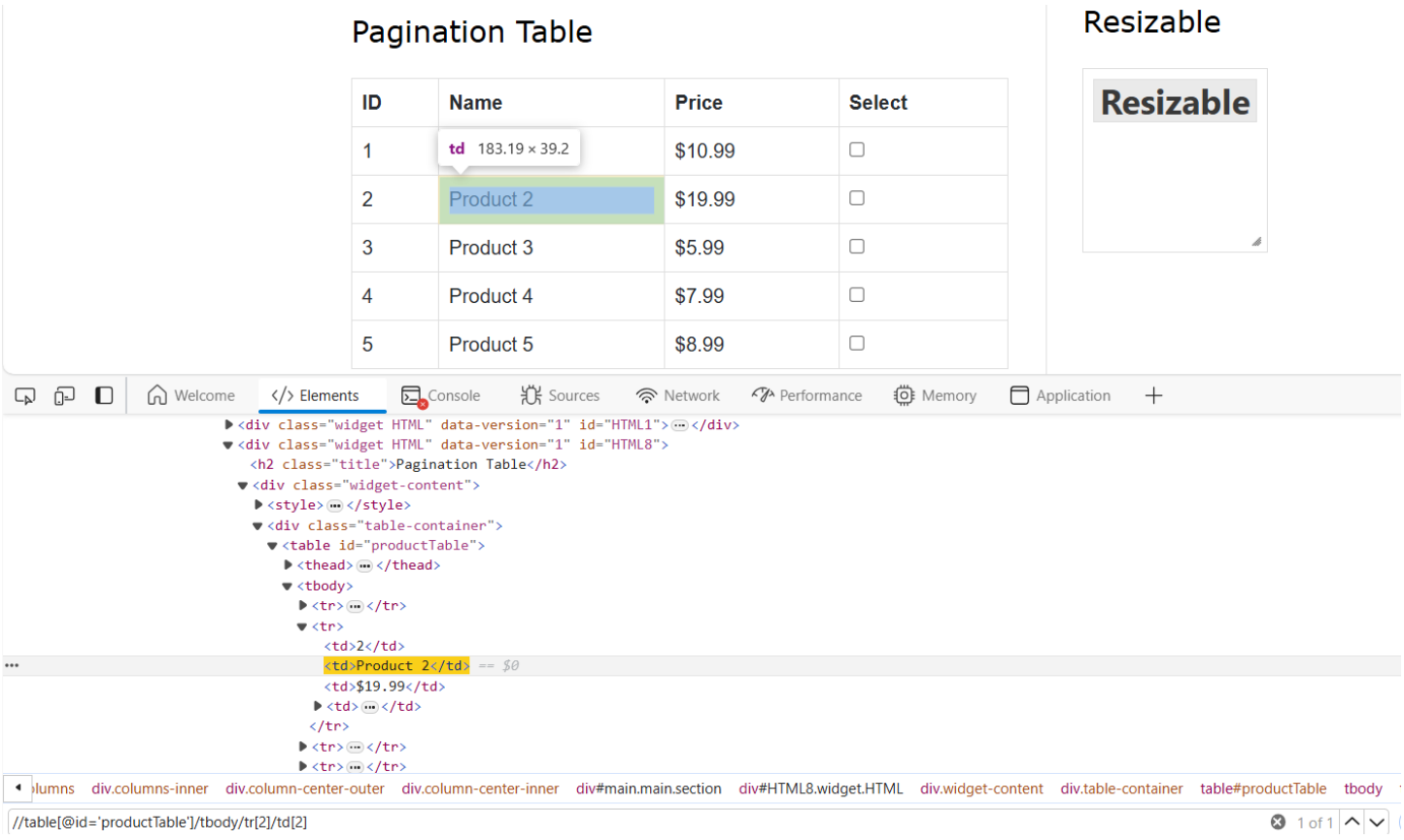



```
<!DOCTYPE html>


<html>
<head>
<style>
table, th, td {

border: 1px solid black;
border-collapse: collapse;
padding: 8px;
.header-color{background-color: grey;}
</style>



</head>
<body>
      <table>
            <thead class="header-color">
                  <tr>
```

```
                        <th>Number</th>
                        <th>First Name</th>
                        <th>Last Name</th>
                        <th>Address</th>
                  </tr>
            </thead>
            <tbody>

                        <tr>
                        <td>1</td>
                        <td>Omar</td>
                        <td>Ramo</td>
                        <td>123 Street</td>
                  </tr>
                  <tr>
                        <td>2</td>
                        <td>John</td>
                        <td>Doe</td>
                        <td>321 Street</td>
                  </tr>
                  <tr>
                        <td>3</td>
                        <td>Jane</td>
                        <td>Doe</td>
                        <td>321 Street</td>
                  </tr>
            </tbody>


      </table>
</body>
</html>
```

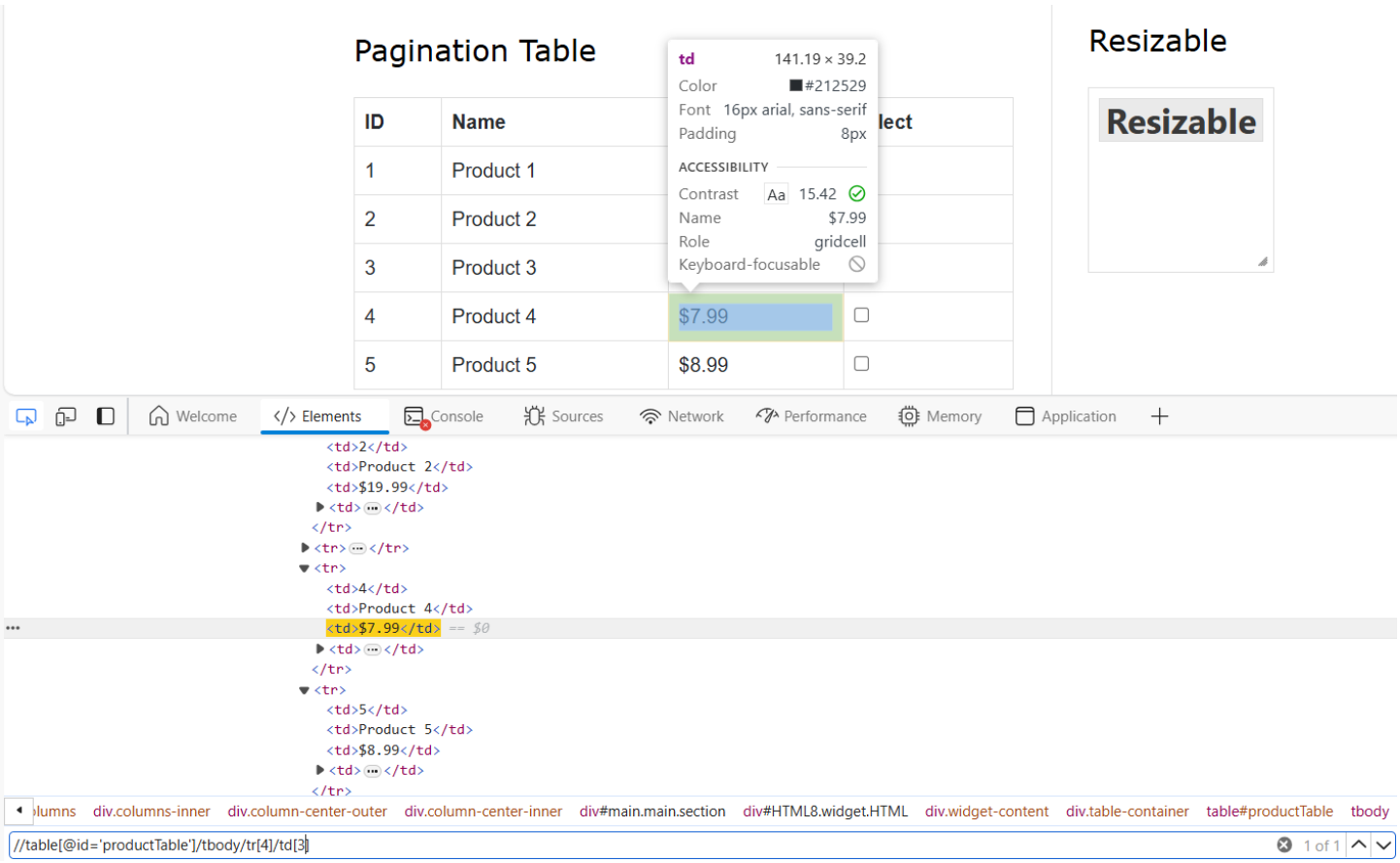
Website for practice:

[Automation Testing Practice (testautomationpractice.blogspot.com)]



//table[@id='productTable']/tbody/tr[2]/td[2]

- `//table[@id='productTable']`: Selects a table element with the attribute `id` equal to `productTable`.
- `/tbody`: Selects the `tbody` child of the table.
- `/tr[2]`: Selects the second `tr` (table row) within the `tbody`.
- `/td[2]`: Selects the second `td` (table cell) within the second row.



//table[@id='productTable']/tbody/tr[4]/td[3]

- `//table[@id='productTable']`: Selects a table element with the attribute `id` equal to `productTable`.
- `/tbody`: Selects the `tbody` child of the table.
- `/tr[4]`: Selects the fourth `tr` (table row) within the `tbody`.
- `/td[3]`: Selects the third `td` (table cell) within the fourth row.

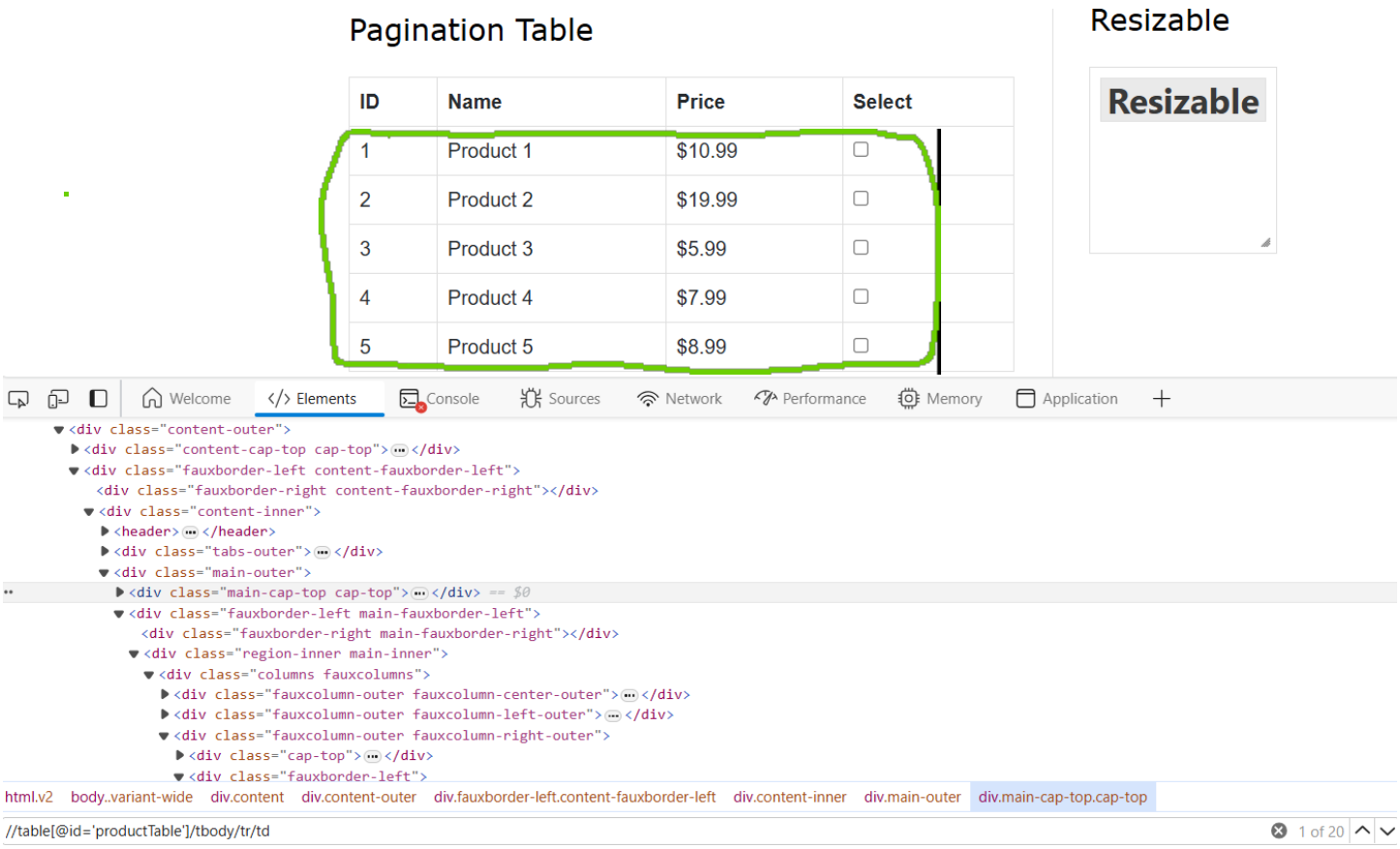
```
package module2;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.List;
public class WebTablesPractice {
      public static void main(String[] args) throws InterruptedException {
// Create a web driver instance
            WebDriver driver = new ChromeDriver();
// driver.manage().window().maximize();
// navigate to the targeted web page
            driver.get("https://testautomationpractice.blogspot.com/");
            Thread.sleep(3000);
// Get the data from product table
// 2 row 2 cell
```

```
            WebElement secondCellData = driver.findElement(By.xpath("//*[@id =
'productTable']/tbody/tr[2]/td[2]"));

// Extract the content text from the element  getText() returns a string
            String secondCellText = secondCellData.getText();
            System.out.println("The data in the second row and second cell is:" +
secondCellText);
            //EXTRACTING DATA USING THE METHOD : getDataByRowAndColumn()
            // navigate to the targeted web page
            driver.get("https://testautomationpractice.blogspot.com/");
            Thread.sleep(3000);
            // Get the data from product table
            // 2 row 2 cell
            getDataByRowAndColumn(driver, 2, 2);
            // Get the data from product table
            // 2 row 3 cell
            getDataByRowAndColumn(driver, 2, 3);
            // Get the data from product table
            // 3 row 1 cell
            getDataByRowAndColumn(driver, 3, 1);
            //get the data from the second column
            List<WebElement> columnData = driver.findElements
            (By.xpath("//*[@id='productTable']/tbody/tr/td[2]"));
            for (WebElement eachElement: columnData){
              String eachCellData = eachElement.getText();
System.out.println(eachCellData);}

            driver.quit();
      }

      // Reusable method that will take two parameters row and column
      // Those params data will be used to build a dynamic xpath
      // The rule of static method: we cannot call directly any non static members
      public static void getDataByRowAndColumn(WebDriver driver, int row, int column) {
            String dynamicXpath = "//*[@id='productTable']/tbody/tr[" + row + "]/td[" +
column + "]";
            WebElement cellData = driver.findElement(By.xpath(dynamicXpath));
            // Extract the content text from the element - getText() returns a string
            String cellText = cellData.getText();
            System.out.println("The data in the row " + row + " and " + column + "second
cell is: " + cellText);
      }

}
```

//table[@id='productTable']/tbody/tr/td

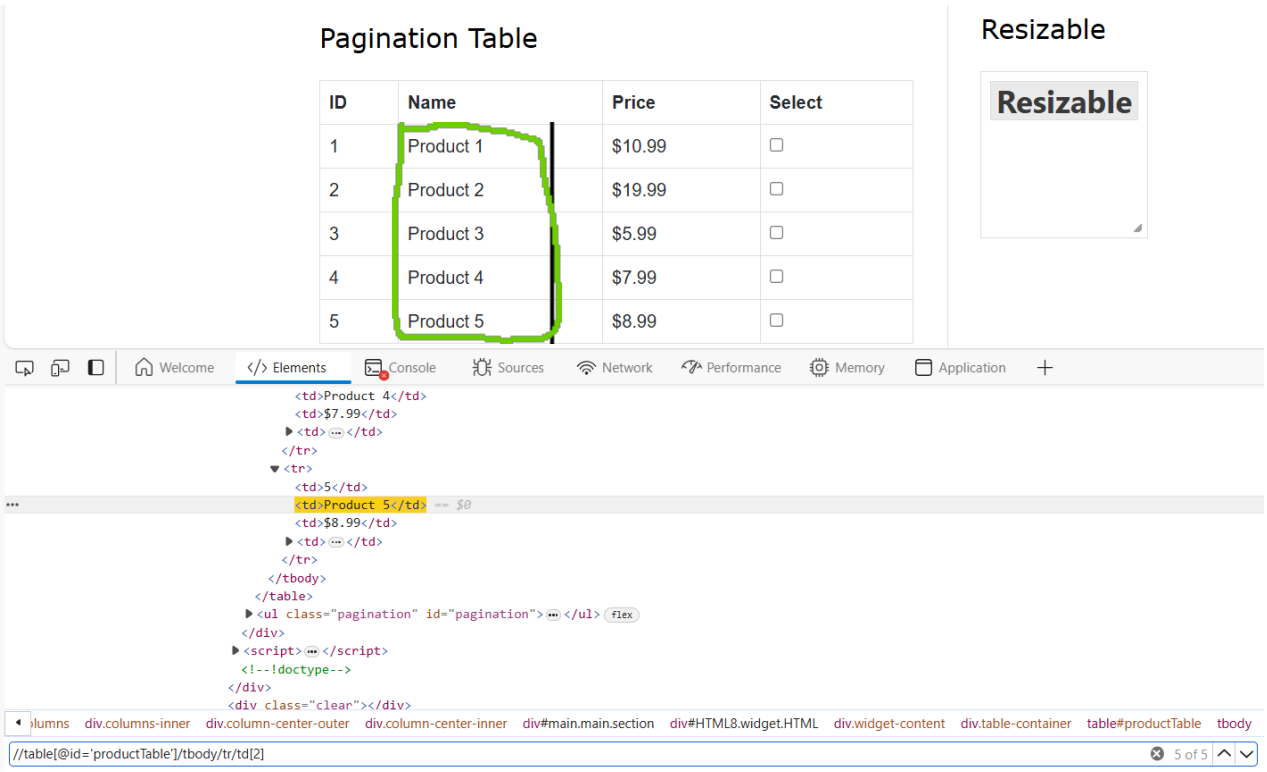
Selects a `<table>` element with the `id` attribute equal to `productTable`.

- `/tbody`: Selects the `<tbody>` child of the table.
- `/tr`: Selects **all** `<tr>` (table row) elements within the `<tbody>`.
- `/td`: Selects **all** `<td>` (table cell) elements within each row.



//table[@id='productTable']/tbody/tr/td[2]

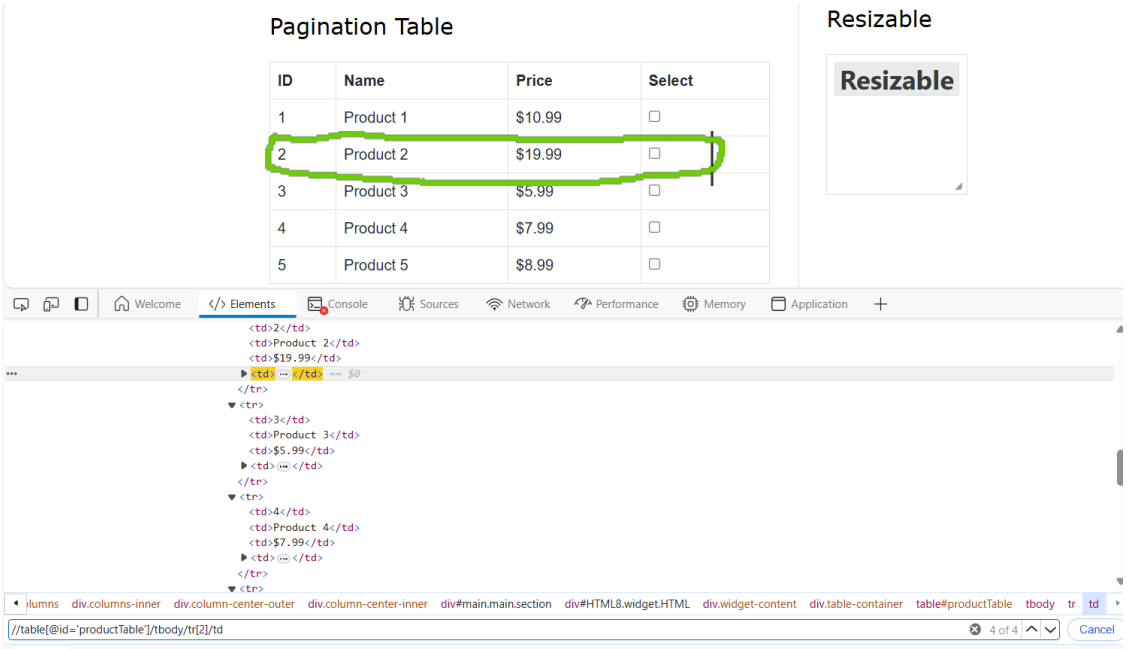
- `/tr`: Selects **all** `<tr>` (table row) elements within the `<tbody>`.
- `/td[2]`: Selects the **second** `<td>` (table cell) in each row.



//table[@id='productTable']/tbody/tr[2]/td

- `/tr[2]`: Selects the **second** `<tr>` (table row) within the `<tbody>`.
- `/td`: Selects **all** `<td>` (table cell) elements within the second row.

```
//get the data from the second column
List<WebElement> columnData = driver.findElements
(By.xpath("//*[@id='productTable']/tbody/tr/td[2]"));
for (WebElement eachElement: columnData){
    String eachCellData = eachElement.getText(); System.out.println(eachCellData);}
```



Rules to indentify a row/column:

1. Locate a specific cell in web table

-> Syntax `//table [@attribute = 'value']/tbody/tr[2]/td[2]` -> `//table[@id = 'productTable']/tbody/tr[2]/td[2]`

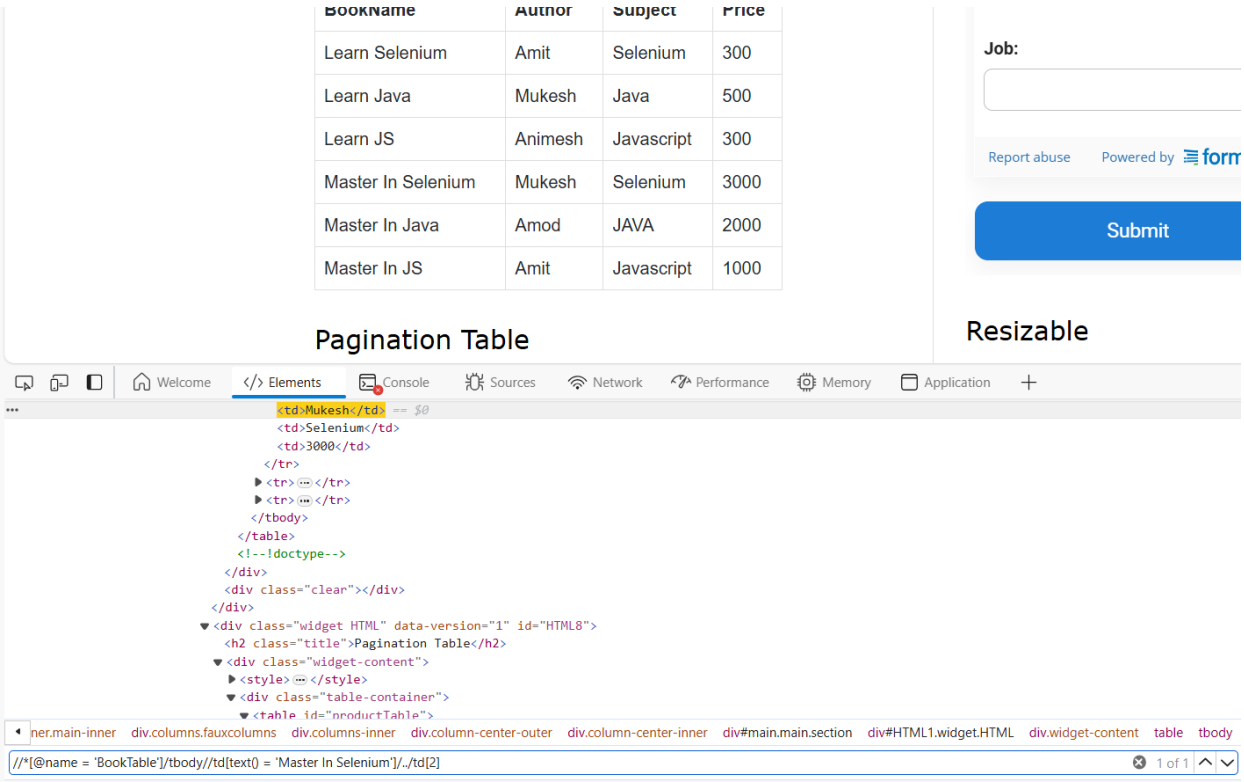
2. Get a whole coumn data:

→> `//table [@attribute = 'value']/tbody/tr/td[columnNumber]`

3. Identify the Book Name Author Dynamically

`//*[@name = 'BookTable']/tbody//td[text() = 'Master In Selenium']/../td[2]`

To indentify **Mokish** we need **first** to locate 'Master In Selenium, **then go back using /..** to locate the second cell.



SAME EXERCISE WITH IDENTIFYING THE CELLS, ROWS, AND COLUMNS:

```
package module2;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

import java.util.List;

public class WebTablesPractice {
      public static void main(String[] args) throws InterruptedException {
// Create a web driver instance
            WebDriver driver = new ChromeDriver();
// driver.manage().window().maximize();
// navigate to the targeted web page
            driver.get("https://testautomationpractice.blogspot.com/");
            Thread.sleep(3000);
// Get the data from product table
// 2 row 2 cell
            WebElement secondCellData = driver.findElement(By.xpath("//*[@id =
'productTable']/tbody/tr[2]/td[2]"));

// Extract the content text from the element  getText() returns a string
            String secondCellText = secondCellData.getText();
            System.out.println("The data in the second row and second cell is:" +
secondCellText);

            // navigate to the targeted web page
            driver.get("https://testautomationpractice.blogspot.com/");
            Thread.sleep(3000);
            System.out.println("Cell data:");
```

```
            // EXTRACTING DATA USING THE METHOD : getDataByRowAndColumn()
            // Get the data from product table
            // 2 row 2 cell
            getDataByRowAndColumn(driver, 2, 2);
            // Get the data from product table
            // 2 row 3 cell
            getDataByRowAndColumn(driver, 2, 3);
            // Get the data from product table
            // 3 row 1 cell
            getDataByRowAndColumn(driver, 3, 1);

            System.out.println("Whole column data:");
            getDataByColumnNumber(driver, "1");
            // get the data from the second column
            getDataByColumnNumber(driver, "2");
            // get the data from the third column
            getDataByColumnNumber(driver, "3");
            // Identify the author name by bookname
            WebElement authorNameElement = driver
                        .findElement(By.xpath("//*[@name = 'BookTable']/tbody//td[text()
= 'Master In Selenium']/../td[2]"));
            String authorName = authorNameElement.getText();
            System.out.println("The Author for Master In Selenium is: " + authorName);

            // Identify the author name by book name
            WebElement authorNameElement2 = driver
                        .findElement(By.xpath
("//*[@name='BookTable']/tbody//td[text()='Master In Java']/../td[2]"));
            String authorName2 = authorNameElement2.getText();
            System.out.println("The Author for Master In Selenium is: " + authorName2);

            driver.quit();
      }

      // Reusable method that will take two parameters row and column
      // Those params data will be used to build a dynamic xpath
      // The rule of static method: we cannot call directly any non static members
      public static void getDataByRowAndColumn(WebDriver driver, int row, int column) {
            String dynamicXpath = "//*[@id='productTable']/tbody/tr[" + row + "]/td[" +
column + "]";
            WebElement cellData = driver.findElement(By.xpath(dynamicXpath));
            // Extract the content text from the element - getText() returns a string
            String cellText = cellData.getText();
            System.out.println("The data in the row " + row + " and " + column + "second
cell is: " + cellText);
      }

      // Create a reusable method that will return the data from the specified column
      // 3 usages
      public static void getDataByColumnNumber(WebDriver driver, String columnNumber) {
            String dynamicXpath = "//*[@id= 'productTable']/tbody/tr/td[" + columnNumber
+ "]";
            List<WebElement> columnData = driver.findElements(By.xpath(dynamicXpath));
            for (WebElement eachElement : columnData) {
                  String eachCellData = eachElement.getText();
                  System.out.println(eachCellData);
            }
```

```
        }
}
```

Homework not done yet

Homework not done yet

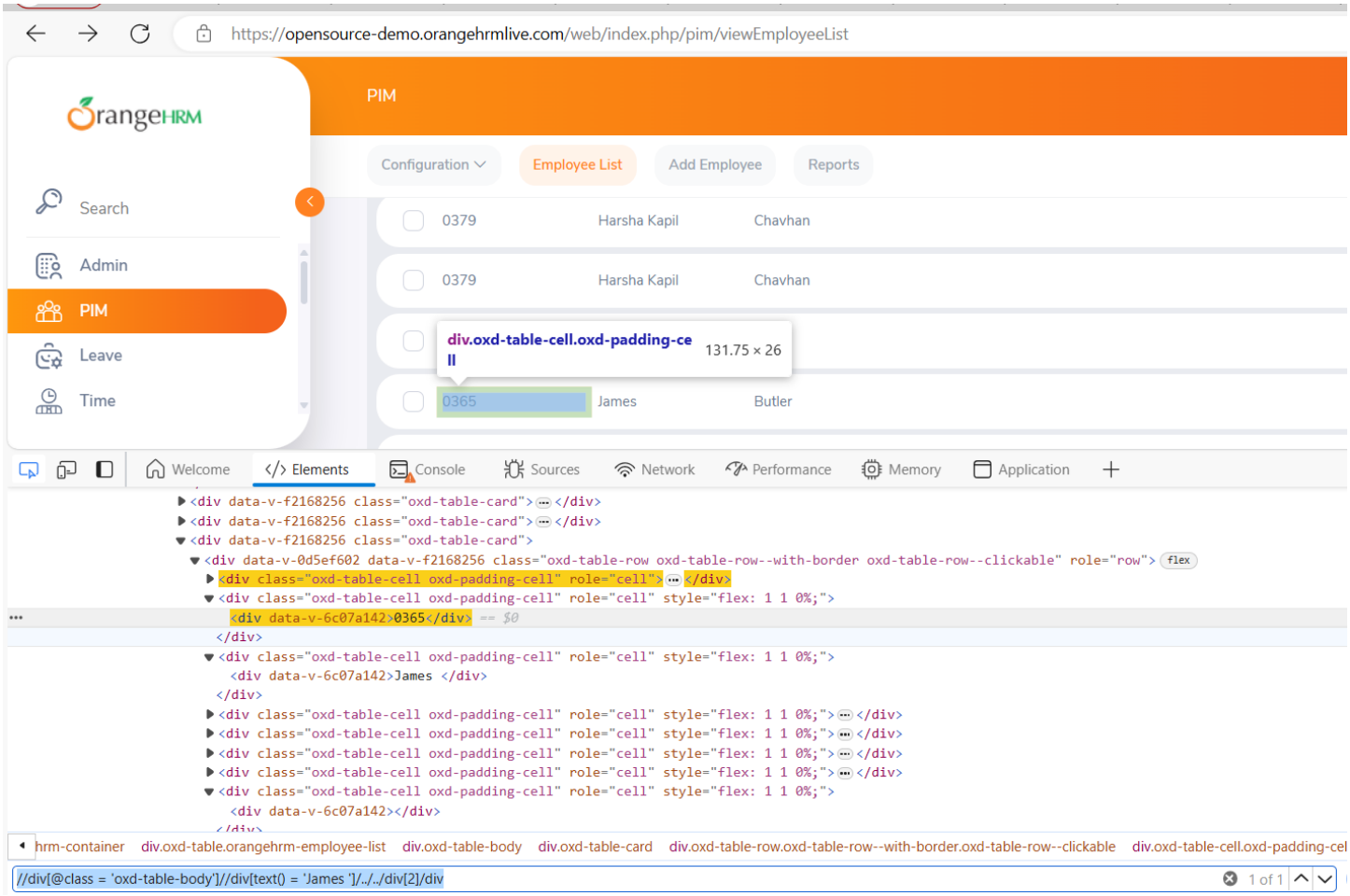
```
Go to https://testautomationpractice.blog spot.com/
Locate the BookTable
And get the suBject/s for author Amit
Homework on tables:
1. On https://testautomationpractice.blog spot.com/
Calculate the total price for ProductTable on the first page.
2. https://us.soccerway.com/national/
england/premier-league/20232024/regular-season/r76443/
```

Homework

The XPath `//div[@class = 'oxd-table-body']//div[text() = 'James ']/../../div[2]/div` is used to locate a specific element in an HTML table-like structure. Here's the breakdown:

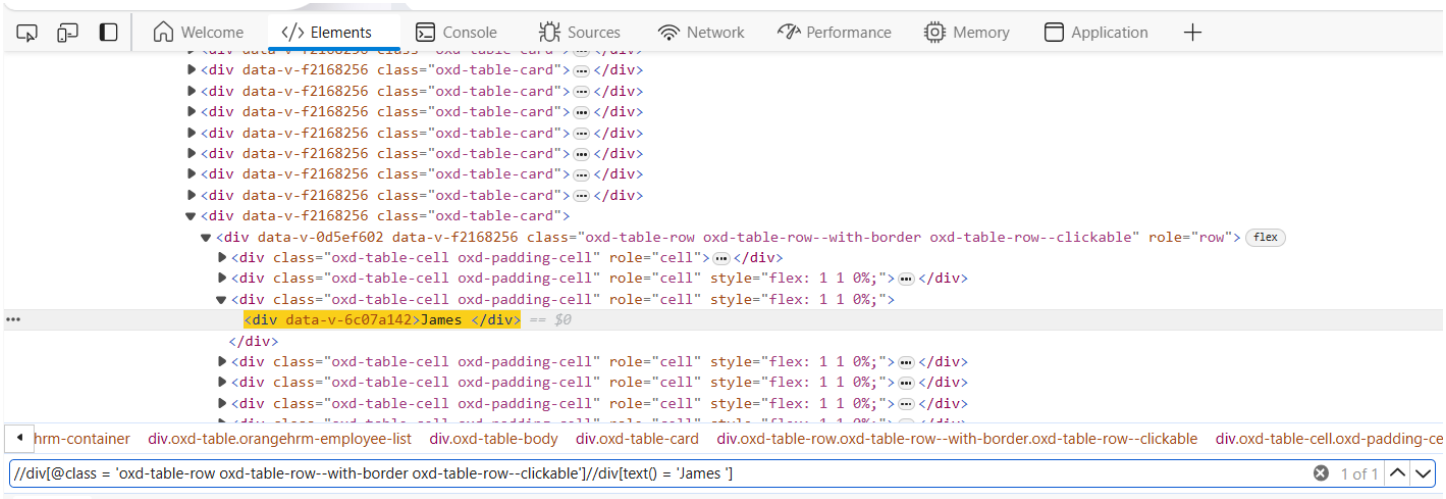
1. `//div[@class = 'oxd-table-body']`: Finds all `div` elements with the class `oxd-table-body`.
2. `//div[text() = 'James ']`: Within the previously selected elements, finds a `div` with the exact text `James` .
3. `/../../div[2]/div`: **Navigates up two levels from the `James` `div`** to its ancestor, then selects the second sibling `div`, and finally gets its child `div`.

This XPath is used to find a specific cell related to the row where `James` is located.



Class value is changing and has a long value divided by space!

we can take the first part, then the second one and put them separately.



//div[@class = 'oxd-table-row oxd-table-row--with-border oxd-table-row--clickable']//div[text() = 'James ']

We can write it like that:

//div[@class = 'oxd-table-body' or @class = 'xd-table-body oxd-card-table-body']//div[text() = 'James ']

CSS LOCATOR:

```
<body>
<div>
      <a href="#link1">Link 1</a>
<p> One </p>
</div>

<div class="link-container">
      <a href="#link2" id="second-link">Link 2</a>
</div>

<div id="third-link-container">
      <a href="#link3">Link 3</a>
<p> Three </p>
</div>
</body>
// Utilizing CSS Selector
1. Locate by tag
a
Selects all a tags
a,p - Selects all a and p tags
// By.tagName("a")
// By.xpath("//a")
// By.cssSelector("a")
2. Select an elemnt with a specific attribute value
xpath> //a[@id='second-link']
cssSelector -> a[id = second-link]
if we have a value that is only one word we don't need quotation marks, it is optional if
there are more than one words than we need to wrap it up in quotes

3. Using wild card
xpath> //*[@id='second-link']
cssSelector -> [id= second-link]

<div title="unique title">
      <div>
            First gen
            <div>
                  Second Gen
                  <div>
                        Third gen
                        <p>Third descendant div</p>
                  </div>
            </div>
      </div>
</div>
```

4.How do we navigate to child elements?

xpath -> //*[@title = 'unique title']/div/div/div/p

cssSelector allows you to move forward on generations: **>** angle bracket (more than) will allow css, to navigate to immidiate child

cssSelector -> [title = 'unique title']>div>div>div>p

Go to 3rd generation of div -> [title= 'unique title']>div>div>div

5.How do we skip generations, go to decendants

xpath -> //*[@title= 'unique title']//p

css -> [title = 'unique title'] p -> **space** here means go to any tag that matches the condition

= -checks equality

*= -checks if contains

^= - checks if starts with

$= - checks if ends with

```
<body>
      <div class="link-container"></div>
</body>
<button id="example-button">
      <a href="#example" name="long-name-example-link">Click me!</a>
</button>
```

6. How do we check if a attribute contains a value

in css we do not have function contains()

a[name *= 'name-example']

7. How do we check if a attribute starts with a value

a[name ^='long-name']

8. How do we check if a attribute ends with a value

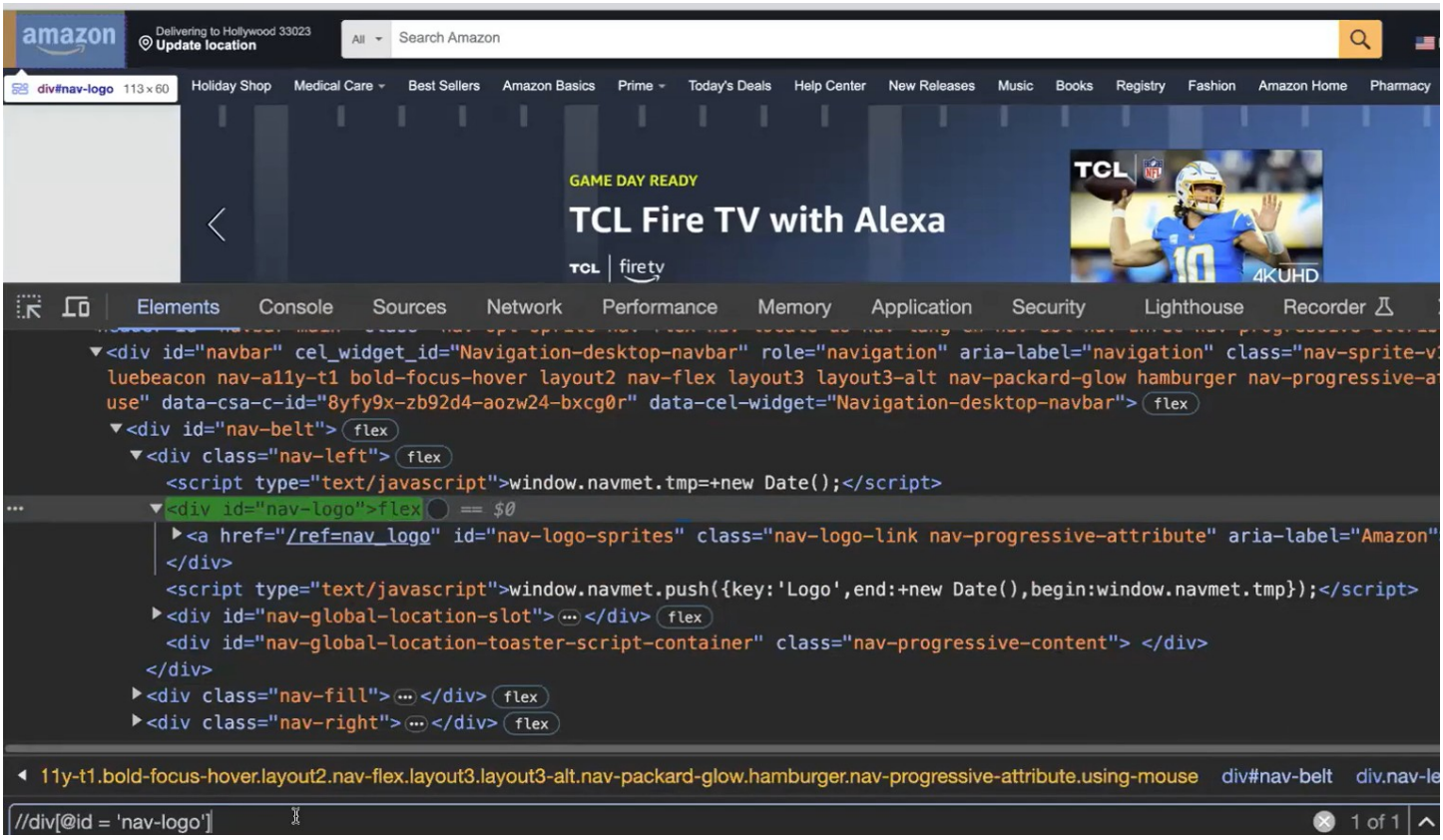
a[name $= 'example-link']

```
/*
**** Creating CSS Selector Using Attributes ****
*  Syntax: //TagName[Att1=Value]
*  In case Att1 is 'id' or 'class' then we can use following symbols:
*      id => # (hash)
*      class => . (dot)
*  Examples: 1. div#u123   2. span.layerParent
*/


/*
* Creating CSS with Multiple Attributes
*
* 1. ANDING the Attributes (No Symbol Required)
*    Syntax: TagName[Att1=Value][Att2=Value]
*
* 2. ORING the Attributes (',' will work as 'OR')
*    Syntax: TagName[Att1=Value], TagName[Att2=Value]
*/
```

```
/*
 * Creating CSS Selectors for Dynamic Elements
 *
 * 1. Using partial value of attribute (Symbol - *)
 *    Example: TagName[Att1*='Partial Value']
 *
 * 2. Using starting value of attribute (Symbol ^)
 *    Example: TagName[Att1^='Starting Value']
 *
 * 3. Using ending value of attribute (Symbol $)
 *    Example: TagName[Att1$='Ending Value']
 */
```

```
/*
 * Advanced CSS Selectors
 *
 * 1. Using only TagName and Attribute Name
 *
 * 2. Node1,Node2 => Matches all Node1s and Node2s
 *
 * 3. Node1 Node2 => Matches all Node2s which are descendant of Node1
 *
 * 4. Node1>Node2 => Matches all Node2s which are children of Node1
 *
 * 5. Node1+Node2 => Matches sibling Node2 placed immediately after Node1
 *
 * 6. Node1~Node2 => Matches all sibling Node2s placed after Node1
 *
 */
```

CSS and XPATH of Amazon logo :



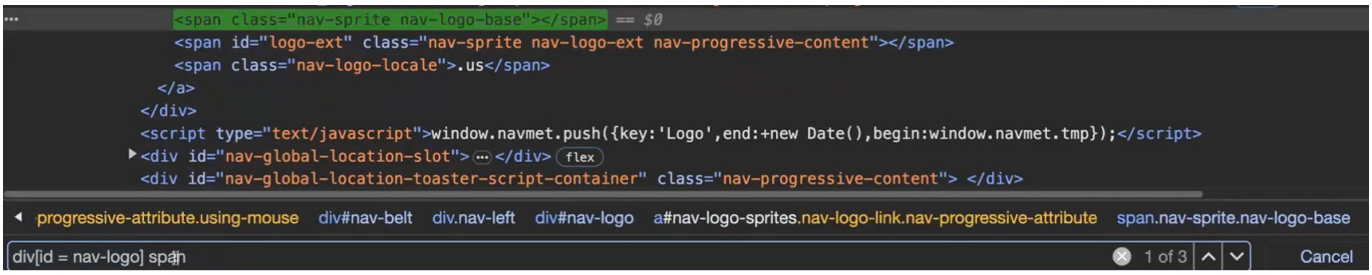
Xpath: //div[@id = ‘nav-logo’]

CSS: div[id = nav-logo]

To go to the children:

CSS: div[id = nav-logo]>a

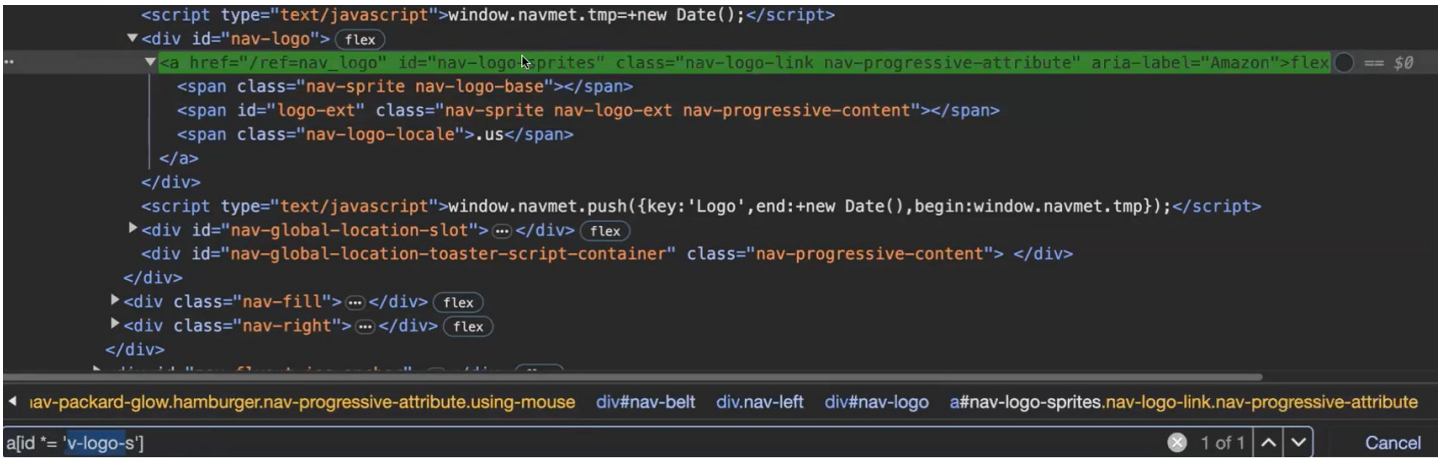
To Skip a couple of generations: put a space



Xpath: //div[@id = nav-logo]//span

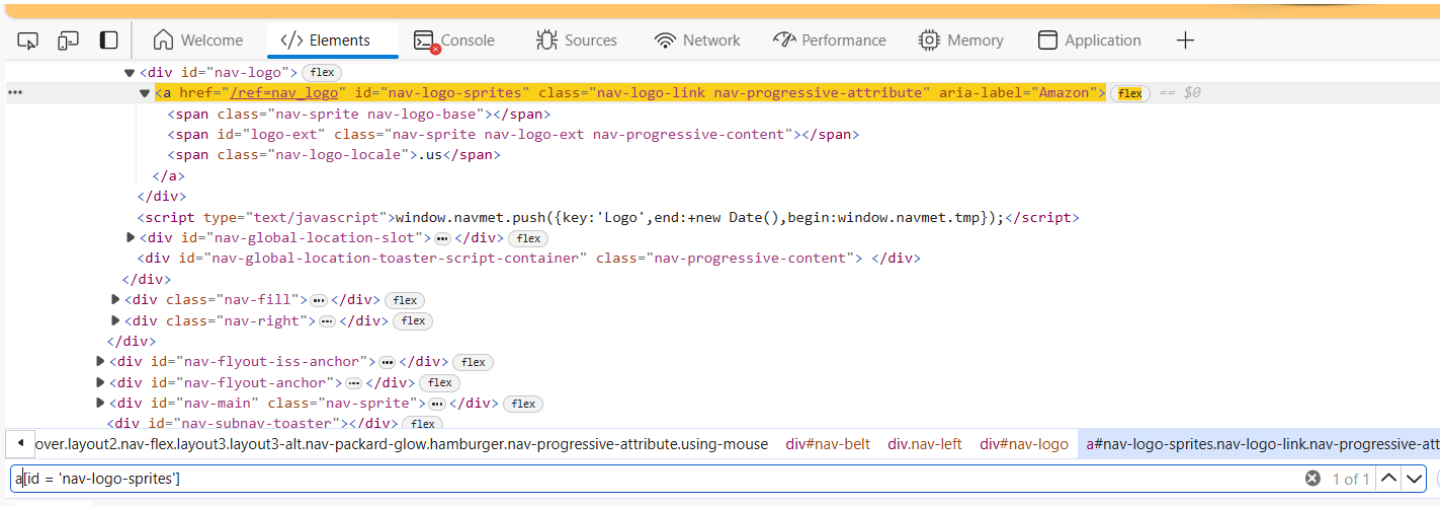
CSS: div[id = nav-logo] span

In CSS we cannot go backward to the parents like xpath.



XPATH: //a[contains(@id, 'v-logo-s')]

CSS: a[id ***=** 'v-logo-s']



XPATH: //a[@id = 'nav-logo-sprites']

CSS : a[id **=** 'nav-logo-sprites']

XPATH: //a[starts-with(@id, 'nav-logo-sprites')]

CSS : a[id $=’logo-sprites’]

XPATH andCSS Coding Summary:

```
Xpath                              VS                          cssSelector
//a                                                           a
//tagName[@attribute = 'value']                               tagName[attribute = value]
//*[@name = 'btn']                                            [name=btn]
//*[@name = 'btn']/input/a                                    [name=btn]>input>a
//*[@placeholder = 'text value']//a                           [placeholder = 'text value'] a
//*[contains(@name, 'feren')]                                 [name *= feren]
//*[strats-with(@name, 'ref')]                                [name ^= ref]
//*[ends-with(@name, 'nce')]                                  [name $= nce]
//input[text() = 'button']                                    not possible to locate by tag text
//*[@class = 'list']/..                                       not possible to navigate backwards

Utilizing Shortcuts
(.) -> stands for class
(#) -> stands for id

Xpath                              cssSel                       cssShortcut
//*[@class = 'button']             [class = button]             .button
//input[@class = 'btn']            input[class = btn]           input.btn
//a[@id = 'btn']                   a[id = btn]                  a#btn

xpath
//div[@id = 'btn']/a[@class = 'container']//input[@name = 'button']
cssSelector
div[id = 'btn']>a[class = 'container'] input[name = 'button']
cssSelector shortcut
div#btn>a.container input[name='button']
```

Indexation in CSS:

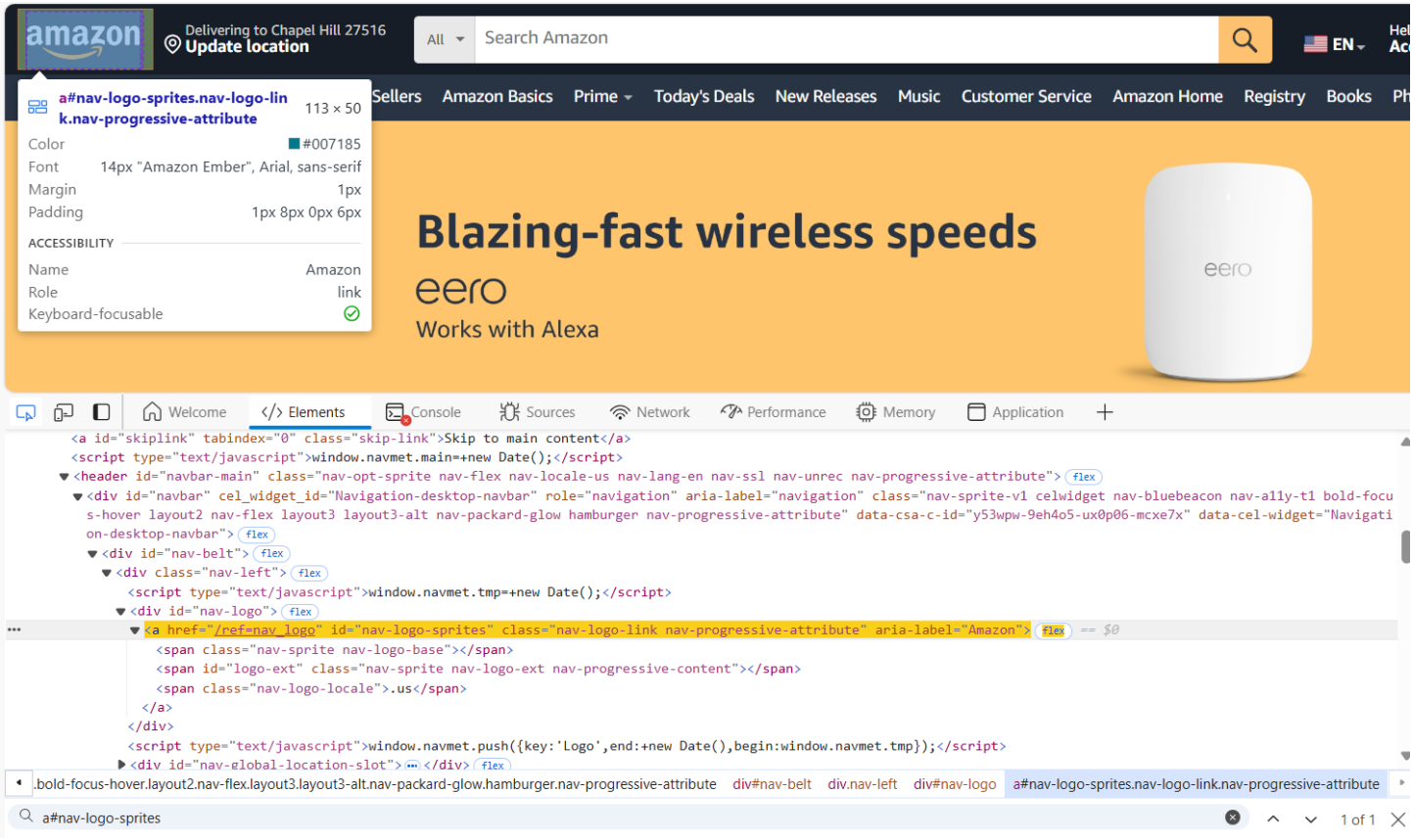
- In Selenium, CSS selectors do not directly support indexing like XPath, but you can use :**nth-child** or :**nth-of-type** pseudo-classes to achieve similar functionality. **Note**: These selectors count from 1, not 0.

There is **no indexation** only when using the **nth-child** or **nth-of-type**:

```
Index in css
(//*[@class = 'textbox'])[4]          -> not possible in cssSelector
//form[@class = 'container']/div[2]   -> form[class = container]>div:nth-child(2)
```

```
Amazon Logo:
a#nav-logo-sprites -> logo
a#nav-logo-sprites>span:nth-child(3) -> contents of logo
```

a#nav-logo-sprites

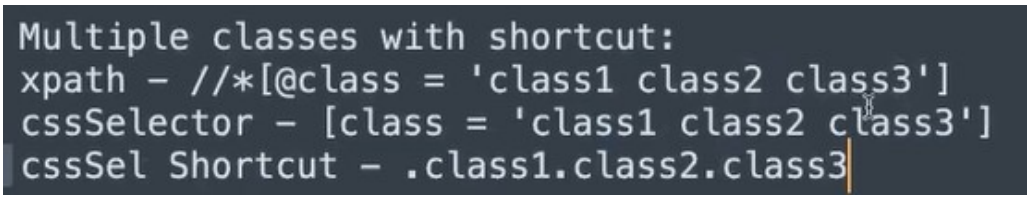


a#nav-logo-sprites>span:nth-child(3)

go forward to the Third span

CSS using Class : a[class='nav-logo-link nav-progressive-attribute']

Using shorcut:


Multiple classes with shortcut:
xpath - //*[@class = 'class1 class2 class3']
cssSelector - [class = 'class1 class2 class3']
cssSel Shortcut - .class1.class2.class3

We need to put a **DOT** instead **of space/spaces**. Orherwise, it will not work:

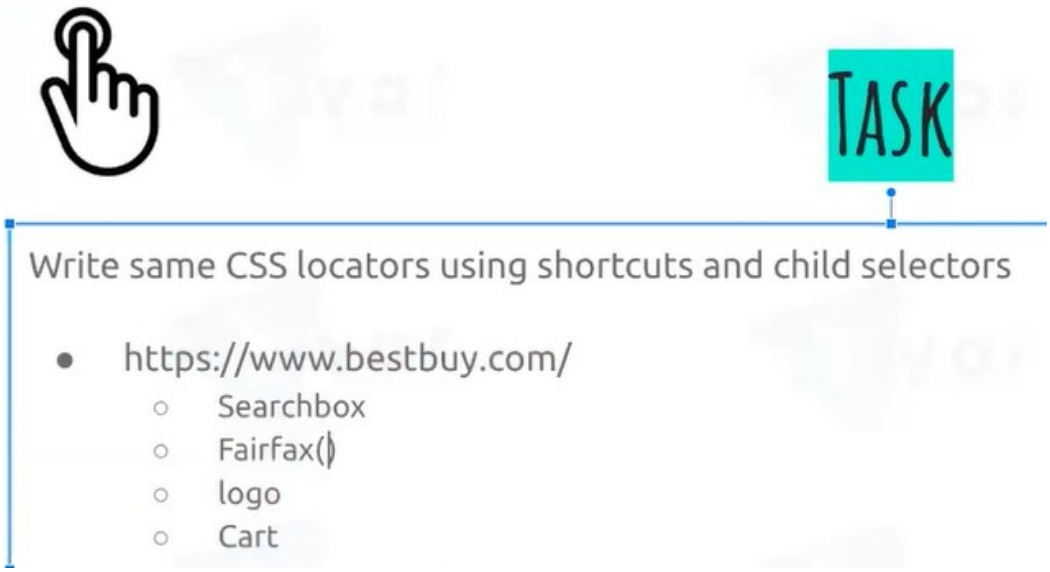
.nav-logo-link.nav-progressive-attribute

~~.nav-logo-link nav-progressive-attribute~~

Using multiple conditions in CSS:

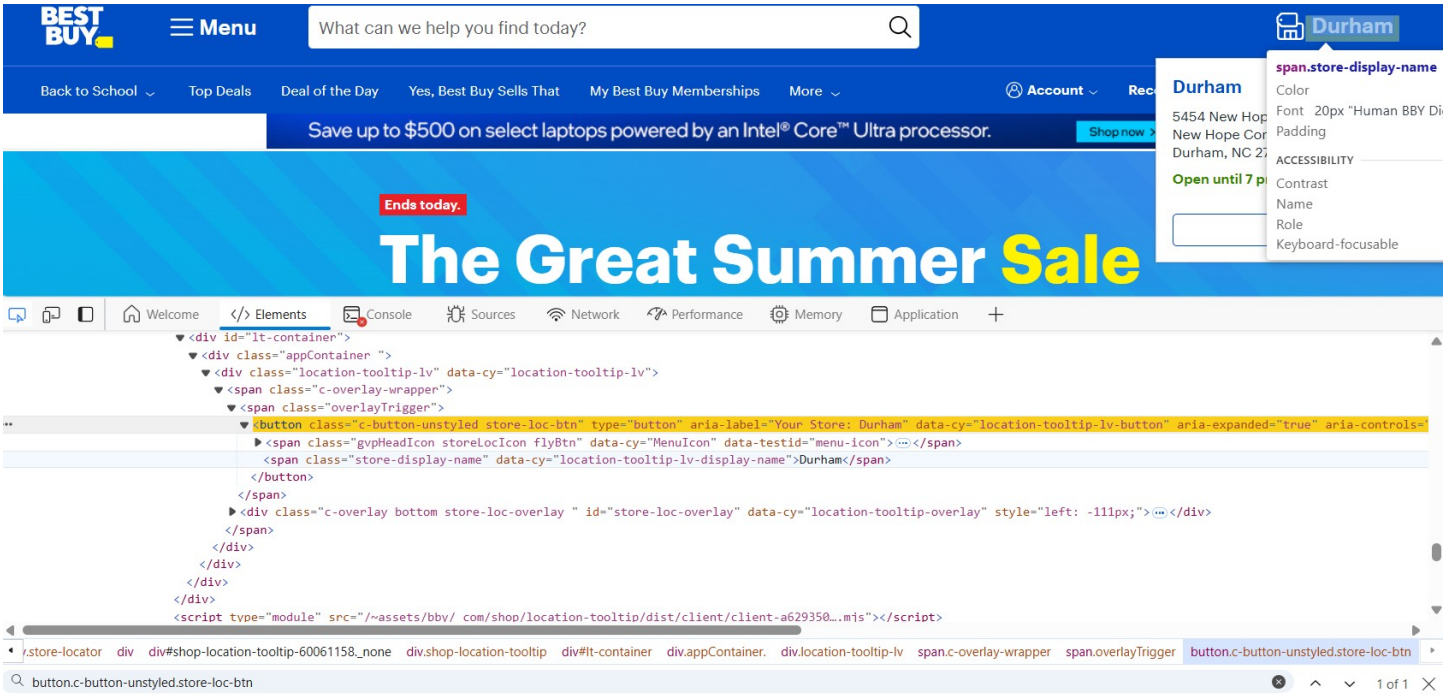
```
<body>
	<div>
		<input name="sampleInput" class="inputClass" type="username">
		<input name="sampleInput" class="inputClass" type="password">
		<input name="sampleInput" class="differentClass" type="text">
	</div>
	<div>
		<input name="anotherInput" class="inputClass" type="username">
		<input name="anotherInput" class="inputClass" type="password">
		<input name="anotherInput" class="differentClass" type="text">
	</div>
</body>
```

```
xpath
//input[@name='sampleInput']->3 elements
cssSelector
input[name='sampleInput']
xpath
//input [@name='sampleInput' and @type='text' and @class='differentClass']
cssSelector AND OPERATOR IN CSS IS EXPRESSED BY A SPACE
input [name = sampleInput] [type = text] [class=differentClass]
<body>
      <div>
            <input id="sampleInput" class="inputClass" type="username">
            <input id ="sampleInput" class="inputClass" type="password">
            <input id ="sampleInput" class="differentClass" type="text">
      </div>
      <div>
            <input id ="anotherInput" class="inputClass" type="username">
            <input id ="anotherInput" class="inputClass" type="password">
            <input id ="anotherInput" class="differentClass" type="text">
      </div>
</body>


xpath
//*[@id='sampleInput' and @class = 'differentClass']
CssSelector AND IN XPATH= SPACE
[id=sampleInput] [class=differentClass]
cssSelector shortcuts
#sampleInput.differentClass
xpath
//input [@type = 'username' or @type = 'password' and @name = 'sampleInput']
CSS: USING OR OPERATOR
not possible to use OR logical operator
```



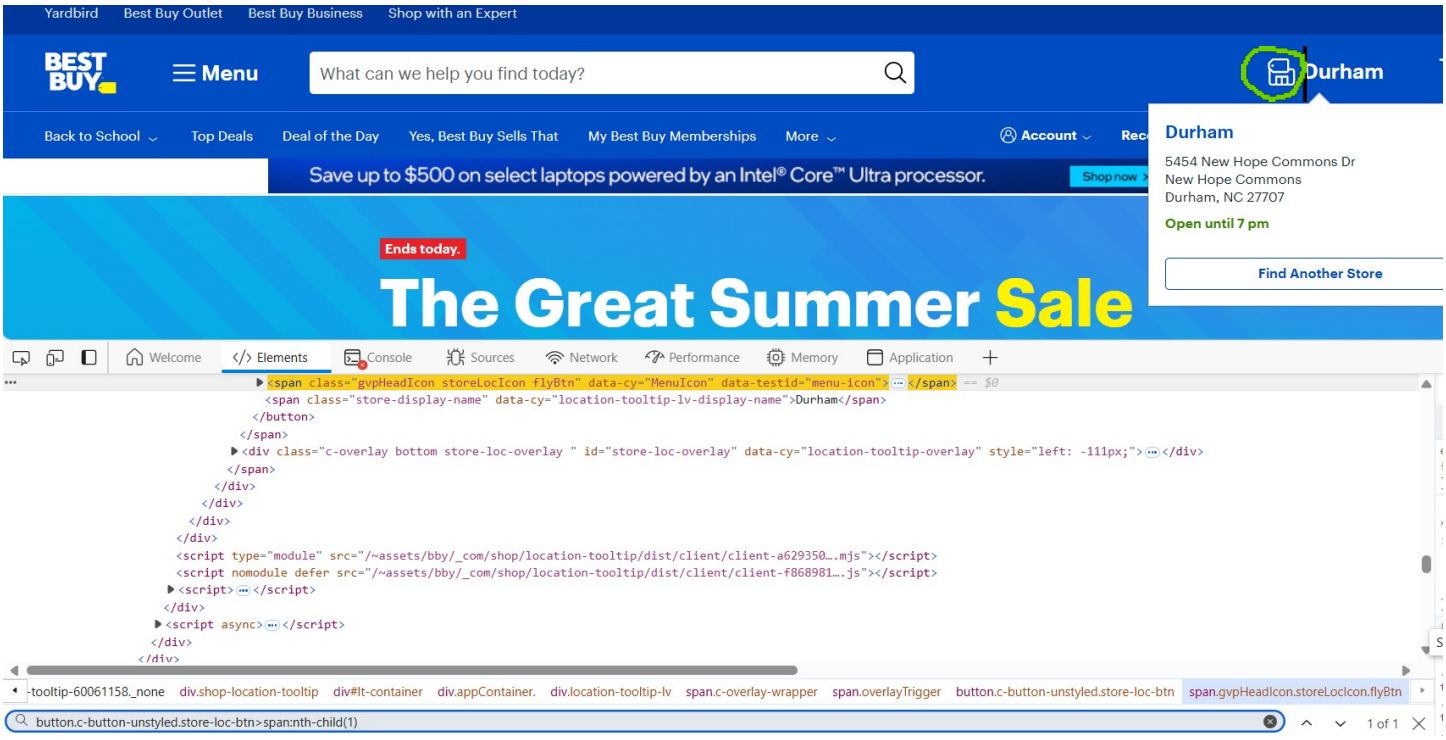
Durham Button

button.c-button-unstyled.store-loc-btn

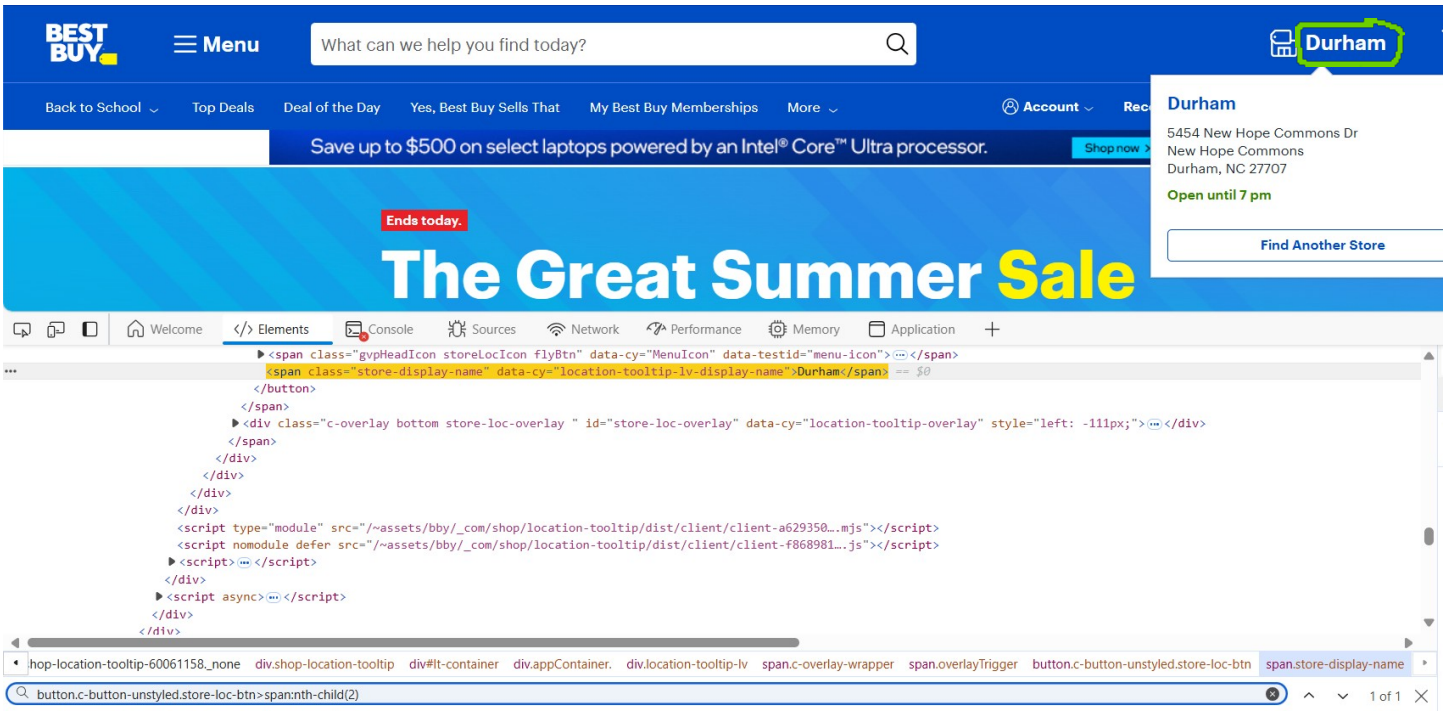


If we choose the first span:

button.c-button-unstyled.store-loc-btn: >span:nth-child(1)



The second span: button.c-button-unstyled.store-loc-btn: >span:nth-child(2)

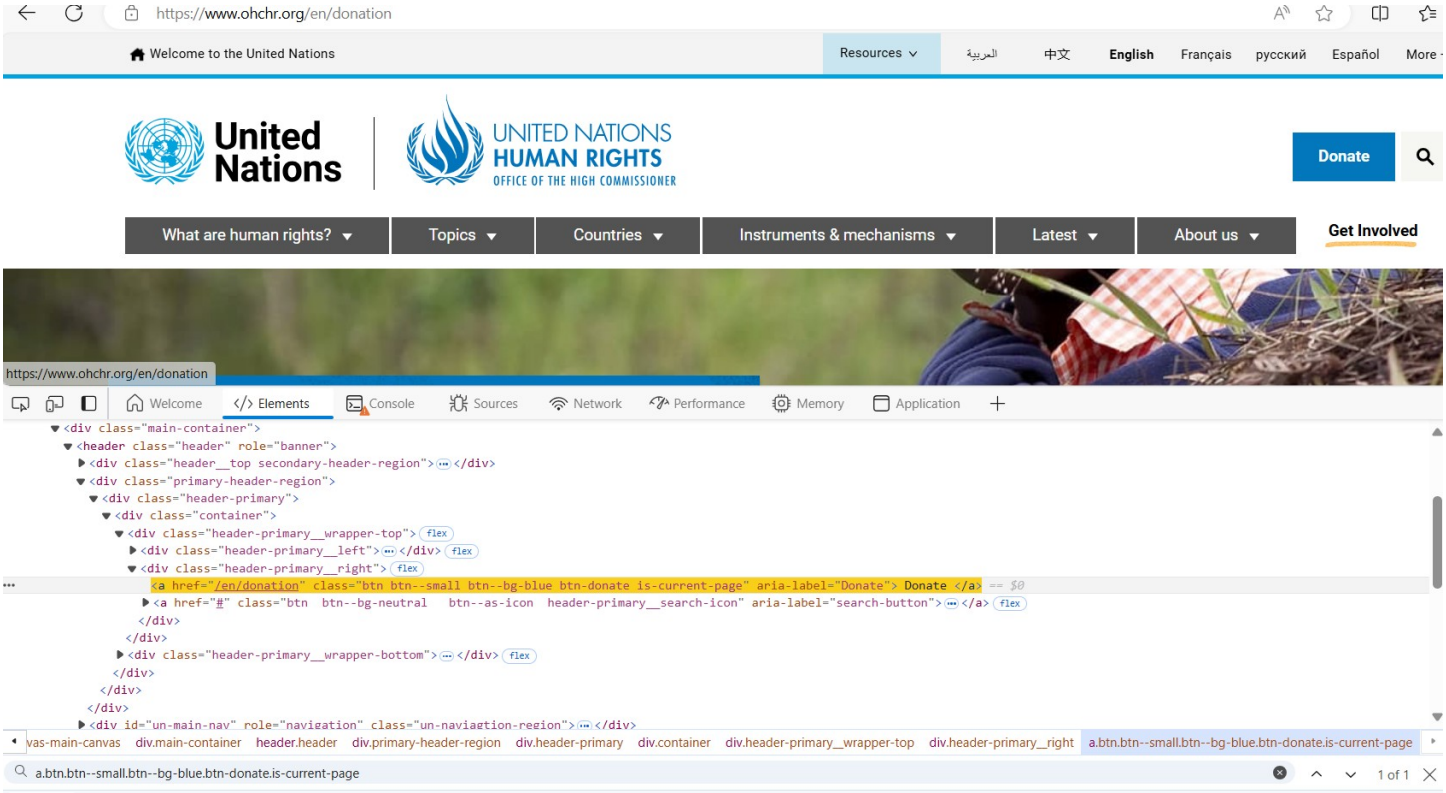


WILDCARDS IN CSS

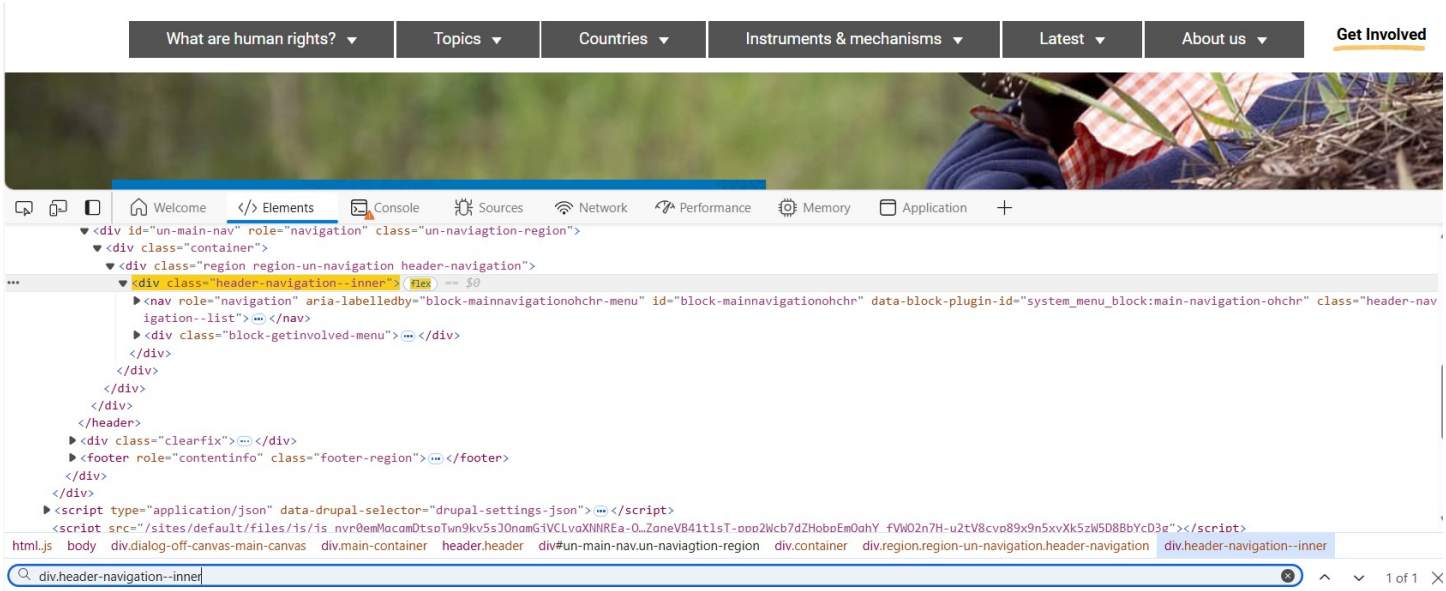
- Elements with multiple class names you can list them as below:
  `driver.findElement(By.css(".submit.primary-btn"));`
- **Child vs Descendant** when working with parent child
  Child → `driver.findElement(By.cssSelector(".submit>div"));`
  Descendant → `driver.findElement(By.cssSelector(".submit div"));`

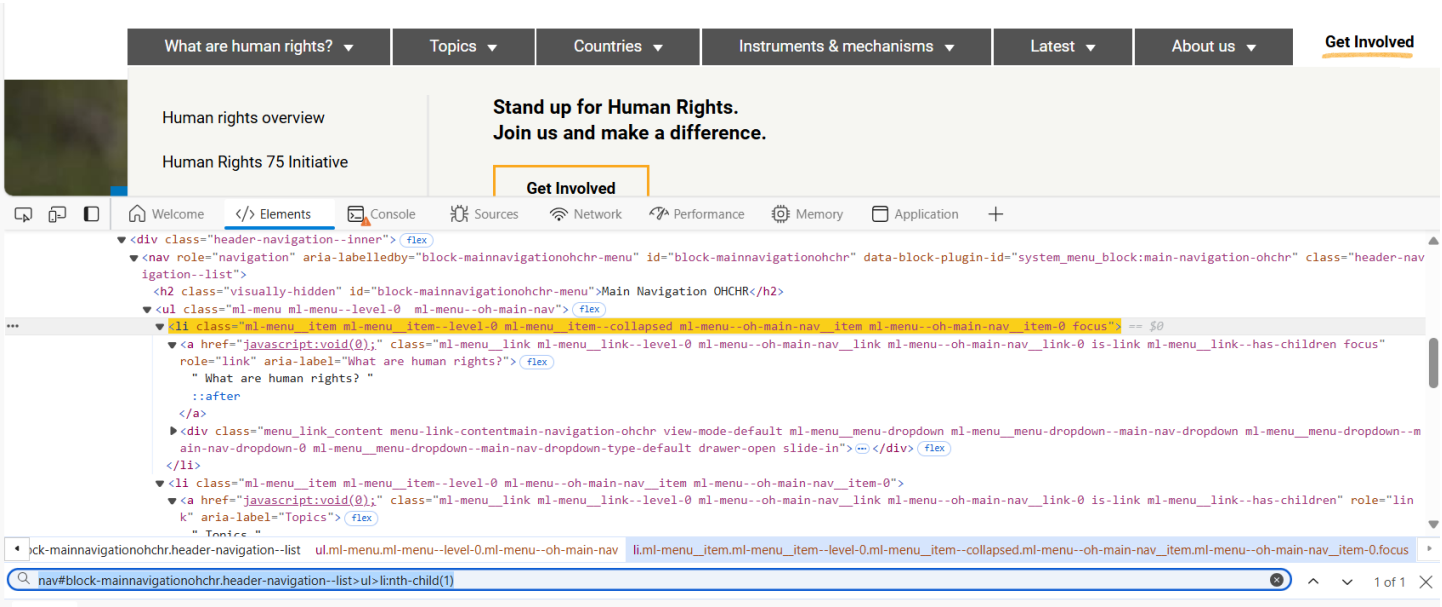
Task

- Visit https://www.ohchr.org/
- Write CSS for Donate Button
- Using CSS locate the navigation bar element
- Then using child elements locate all the elements: Topics, About Us, Countries etc.
  - Same locator will identify 6 elements. You will see it as (1 of 6) in chrome search bar. Not 1 of 1.



a.btn.btn--small.btn--bg-blue.btn-donate.is-current-page





What are human rights ? : **nav#block-mainnavigationohchr.header-navigation--list>ul>li:nth-child(1)**

Topics: **nav#block-mainnavigationohchr.header-navigation--list>ul>li:nth-child(2)**

Countries: **nav#block-mainnavigationohchr.header-navigation--list>ul>li:nth-child(3)**

Instruments & mechanisms: **nav#block-mainnavigationohchr.header-navigation--list>ul>li:nth-child(4)**

Latest: **nav#block-mainnavigationohchr.header-navigation--list>ul>li:nth-child(4)**

About us: **nav#block-mainnavigationohchr.header-navigation--list>ul>li:nth-child(6)**

Task

- Go to url: http://dev-hrm.yoll.io/
- Enter username: yoll-student
- Enter password: Bootcamp5#
- Click login
- Validate the title of the home page
- Validate that the page header is “Dashboard”
- Close the browser.

```
package module2;


/* AC
* Given user navigates to http://dev-hrm.yoll.io/
* Then user enters username voll-student
*And user enters password Bootcamp5#
*And use clicks on login button
*Then prints the title
And the header page is "Dashboard"
**/
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
public class OrangeHRMLoginFlow {
    public static void main(String[] args) throws InterruptedException {
        WebDriver driver = new ChromeDriver();
        // driver.manage().window().maximize();

        driver.get("http://dev-hrm.yoll.io/");
        String username="yoll-student";
        String password = "Bootcamp5#";
        WebElement usernameInput = driver.findElement(By.id("txtUsername"));
        WebElement passwordInput = driver.findElement(By.id("txtPassword"));
        WebElement LoginButton = driver.findElement(By.id("btnLogin"));
// Interact with elements
        usernameInput.sendKeys(username);
        passwordInput.sendKeys(password);
        LoginButton.click();
        Thread.sleep(5000);
        String title = driver.getTitle();
        System.out.println("Page title: " + title);
// create a variable that contains the expected page header
    String expectedPageHeader = "Dashboard";
        WebElement pageHeader = driver.findElement(By.cssSelector(".head>h1"));
        String actualPageHeader = pageHeader.getText();
// Validate if the page header is dashboard
        if (expectedPageHeader.equals (actualPageHeader)){
            System.out.println("TEST: PASSED");
        } else {
            System.out.println("TEST: FAILED");
```

```
        }}

    }
```

FindElement # FindelementS

```
// If you need to verify if an element is present on the page using findElement will be a
bad idea,
// since it will throw an exception when element not found
// Verify that the Yoll logo is present on the homepage after login
        driver.findElement(By.cssSelector("ing[src *= 'yoll-logosssssss']"));
        // exception occured at runtime, at this line
        System.out.println("The logo was located successfully!");
// Instead we can utilize findElements logic

        List<WebElement> logosElement = driver.findElements (By.cssSelector("img[src *=
'yoll-logo']"));

               System.out.println("Located elements:"+ logosElement.size());
               int sizeElement=logosElement.size();
        if (sizeElement>0) {
            System.out.println("Element is present on the page");
        } else {
            System.out.println("Element is NOT present on the page");

        }
                driver.quit();
// findElement vs findElements
// findElement - returns a web element when the locator is correct, otherwise, it throws
an exception
// findElements - returns a list of web elements, if the locator is correct it will
return all element of that locator strategy
//however, if there are no element under that locator, it will return an empty list
```

```
Page title: OrangeHRM
TEST: PASSED
Located elements:1
Element is present on the page


Process finished with exit code 0
```

Results:

FindelementS

When the element is not found:

Located elements:0

Process finished with exit code 0

When the element is found:

Located elements:1

Process finished with exit code 0

Findelement:

When the element is not found:Exception

Exception in thread "main" org.openqa.selenium.NoSuchElementException: no such element: Unable to locate element: {"method":"css selector","selector":"ing[src = 'yoll-logosssssss']"}

(Session info: chrome=127.0.6533.73)

For documentation on this error, please visit: https://www.selenium.dev/documentation/webdriver/troubleshooting/errors#no-such-element-exceptionProcess

finished with exit code **1**

When the element is found:

Process finished with exit code **0**

Summary:

Login Btn I verify that login is present on the page
1. Validate that user can successfully login to the application
When do I want as a tester to verify that a element is actually present on the html page:
- Ui Consistency
- New added feature
Some conditinal points
User accessibility
Worlflow Navigation
2. In selenium we have a method isDisplayed (), that by identifying an existing element on the html if it is present to UI.

3.isDisplayed() will check if the element has any of the styling options as: display: none, visibility: hidden, opacity:All of these are css styling options as: And based on the presence of these value identify is element is displayed or not.

isDisplayed() will return true if none of those stylings are applied false otherwise, We can also use selenium method isEnabled() that will tells us if the element is disabled or not with a boolean value true or false.

isEnabled() checks if a tag has the attribute disabled, if tells you that it is not enabled, otherwise, it will be enabled. isEnabled is an html attribute tag.
4. Getting more information about element by extracting the value of attributes.
Common way of attributes representation is key value format attribute "value"

Occasionally, we can have an attribute as only the key: value, disabled etc...
In **this case** the getAttribute() will **return** black or **null**.
When working with **boolean** attributes, it will **return true if** present **null if** not.

- getAttibute() returns 'true' in case if you are searching for a boolean value
  - These are a few boolean attributes : async, autofocus, autoplay, checked, compact, complete, controls, declare, defaultchecked, defaultselected, defer, disabled, draggable,
- We **cannot** extract text from the textbar or any editable item using **getText()**, to get a text from the editable we have to use **getAttribute("value")**

```
driver.findElement(By.id("item")).getAttribute("value");
```

- Go to Orange HRM
- Maximize window
- Enter below credentials and click to Login Button

User-> yoll-student

Pwd -> Bootcamp5#

- Look for a Welcome Label on the page and return true or false if element is displayed

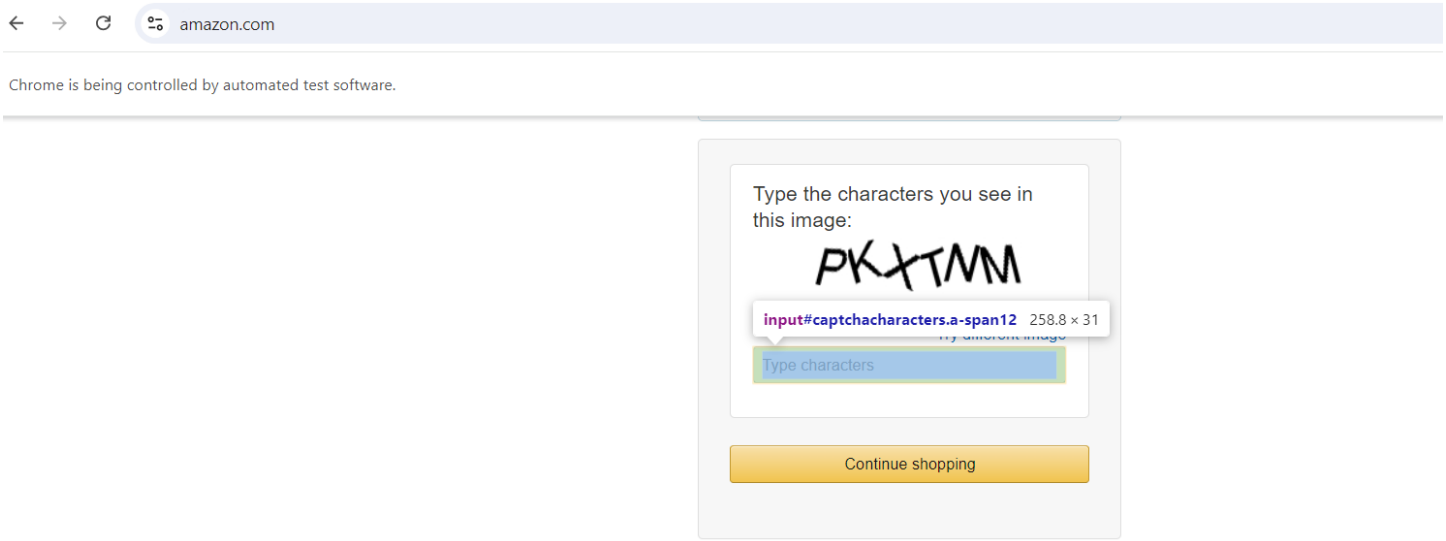
```
package module2;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class practiceIsDisplayed
{

    public static void main(String[] args) throws InterruptedException {
        WebDriver driver = new ChromeDriver();
// setup
// driver.manage().window().maximize();
        Thread.sleep( 2000);
        driver.get("http://dev-hrm.yoll.io");
        String username = "yoll-student";
        String password = "Bootcamp5#";
        WebElement usernameInput = driver.findElement(By.id("txtUsername"));
        WebElement passwordInput = driver.findElement(By.id("txtPassword"));
        WebElement LoginButton = driver.findElement(By.id("btnLogin"));
// Interact with elements
// Print true or false if welcome message is displayed
```

```java
// store web element into a reference/object
        WebElement welcomeLabel = driver.findElement(By.id("welcome"));
        boolean isMessageDisplayed = welcomeLabel.isDisplayed();        // returns true
or false
        System.out.println("The welcome message is displayed:" + isMessageDisplayed);

// Option 2id("welcome")).isEnabled(); System.out.pr
        boolean isMessageDisplayed2 = driver.findElement(By.id("welcome")).isDisplayed();
        System.out.println("Is message displayed: "+isMessageDisplayed2);
        boolean isMessageEnabled = driver.findElement(By.id("welcome")).isEnabled();
        System.out.println("Is message enabled: " + isMessageDisplayed);}}
```

Task :

Got to https://www.amazon.com

verify the placeholder with the test for inputbox of the characters



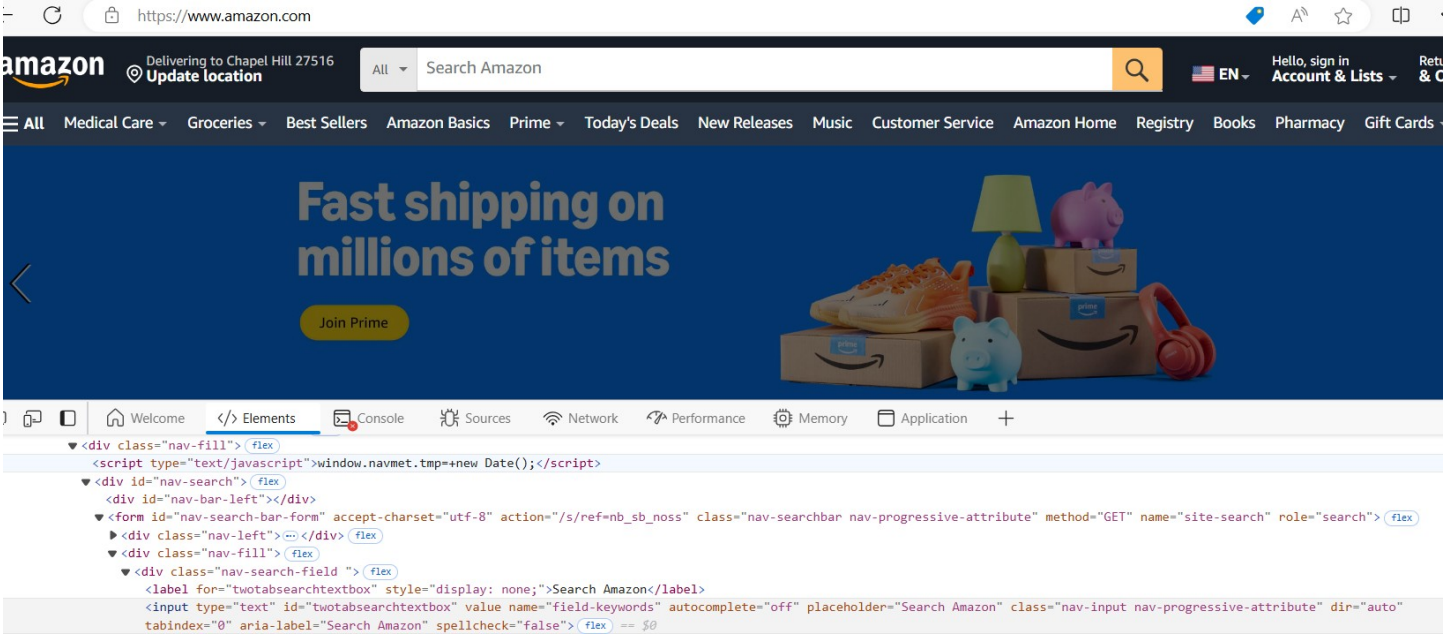
```java
package module2;

import org.openqa.selenium.By;
```

```java
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class getAttributePractice {
    public static void main(String[] args) throws InterruptedException {

        WebDriver driver = new ChromeDriver();
        // Setup
        driver.manage().window().maximize();

        driver.get("https://www.amazon.com/");
        Thread.sleep(2000);

        // Verify that search input box has a placeholder "Search Amazon"
        String expectedPlaceholderValue = "Type characters";

        // Locate the search input box
        WebElement searchInputBox = driver.findElement(By.id("captchacharacters"));

        // Use getAttribute method to get the placeholder value
        String actualPlaceholderValue = searchInputBox.getAttribute("placeholder");

        System.out.println("Placeholder value: " + actualPlaceholderValue);
        if (expectedPlaceholderValue.equals(actualPlaceholderValue)) {
            System.out.println("TEST: PASSED");
        } else {
            System.out.println("TEST: FAILED");
        }


        driver.quit();
    }
}
```

Task :

Got to https://www.amazon.com

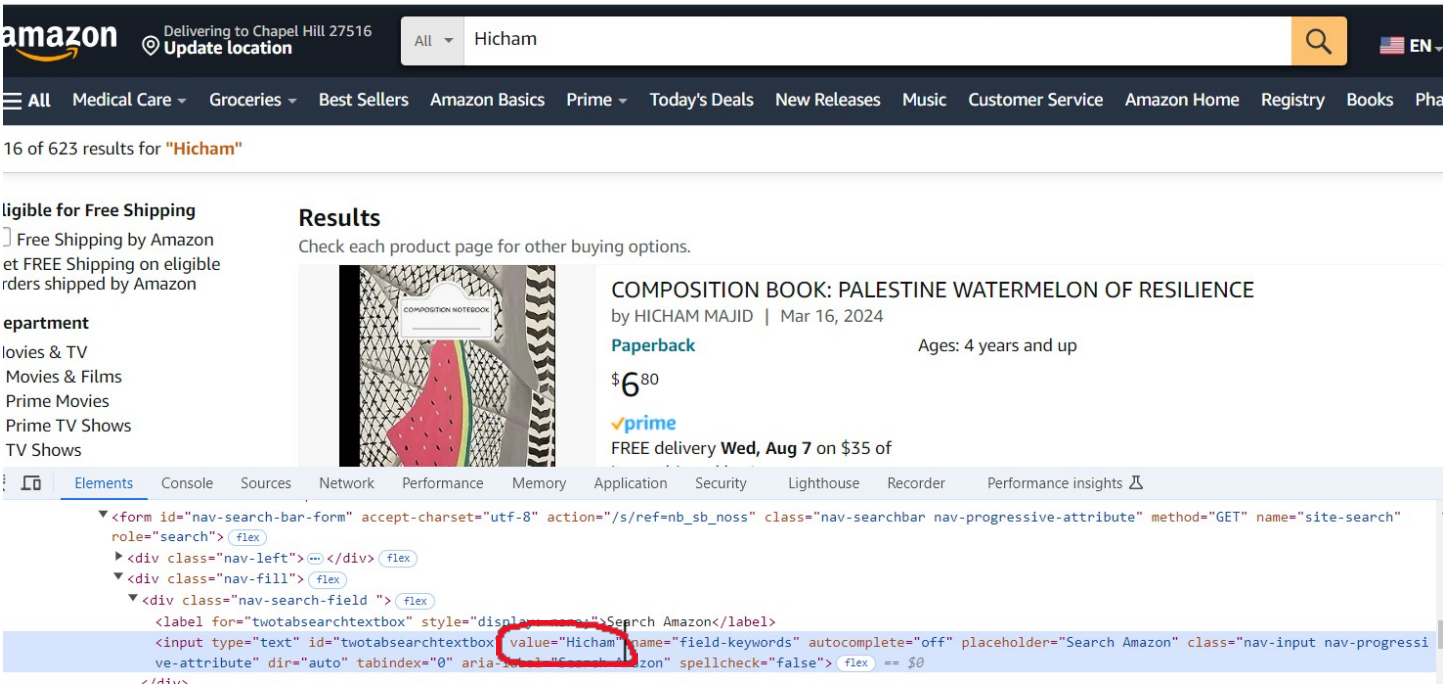
verify the placeholder with the test for inputbox of Search Amazon




```java
package module2;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class getAttributePractice {
    public static void main(String[] args) throws InterruptedException {

        WebDriver driver = new ChromeDriver();
// setup
// driver.manage().window().maximize();

        driver.get("https://www.amazon.com/");
        Thread.sleep(2000);

        // Verify that search input box has a placeholder Search Amazon
        String expectedPlaceholderValue = "Search Amazon";
        // Option 1
        WebElement searchInputBox = driver.findElement(By.id("twotabsearchtextbox"));
        // getAttribute method takes a string parameter as the key or the name of the
attribute we need to
        // the return of this method is string
        String actualPlaceholderValue = searchInputBox.getAttribute("placeholder");

        System.out.println("Placeholder value: " + actualPlaceholderValue);
        if (expectedPlaceholderValue.equals(actualPlaceholderValue)) {
            System.out.println("TEST: PASSED");
        } else {

            System.out.println("TEST: FAILED");
        }

        // Option 2:
       //String placeholder
=driver.findElement(By.id("twotabsearchtextbox")).getAttribute("placeholder");
```

```
        // Let's extract the value that user enters in search bar
        System.out.println("Value in the search bar before sending keys: " +
searchInputBox.getAttribute("value"));



// Get the value of a boolean attribute disabled
System.out.println("Value for disabled attribute is: + searchInputBox.getAttribut"+
("disabled"));

        driver.quit();
    }
}
```

When the user enters a value in the search box, the VALUE ATTRIBUTE GETS A THE value entered



Example 2

```
// Let's extract the value that user enters in search bar

System.out.println("Value in the search bar before sending keys: " +
searchInputBox.getAttribute("value"));

        searchInputBox.sendKeys( "iPhone 15 and laptop bla bla");
        searchInputBox.click();
        System.out.println("Value in the search bar after sending keys:"+
searchInputBox.getAttribute("value"));
```

```
Placeholder value: Search Amazon
TEST: PASSED
Value in the search bar before sending keys:
Value in the search bar after sending keys: iPhone 15 and laptop bla bla
Value for disabled attribute is: null

Process finished with exit code 0
```

If disabled does not exist, it returns nothing:

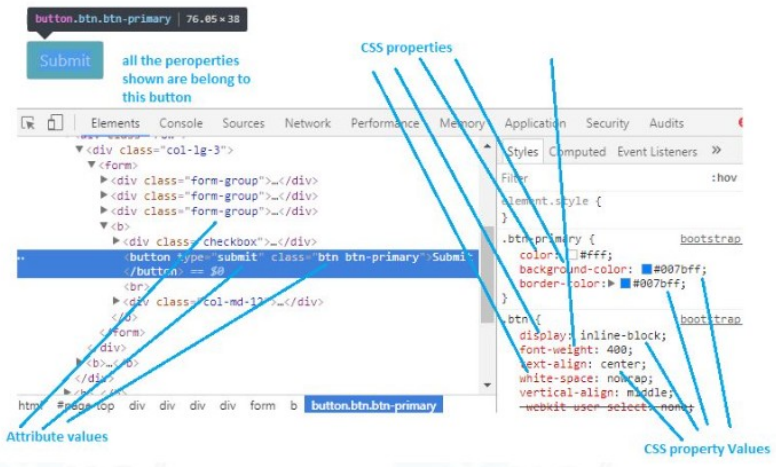
```
// Get the value of a boolean attribute disabled
System.out.println("Value for disabled attribute is: + searchInputBox.getAttribut"+
("disabled"));
```

```
Value for disabled attribute is: null

Process finished with exit code 0
```

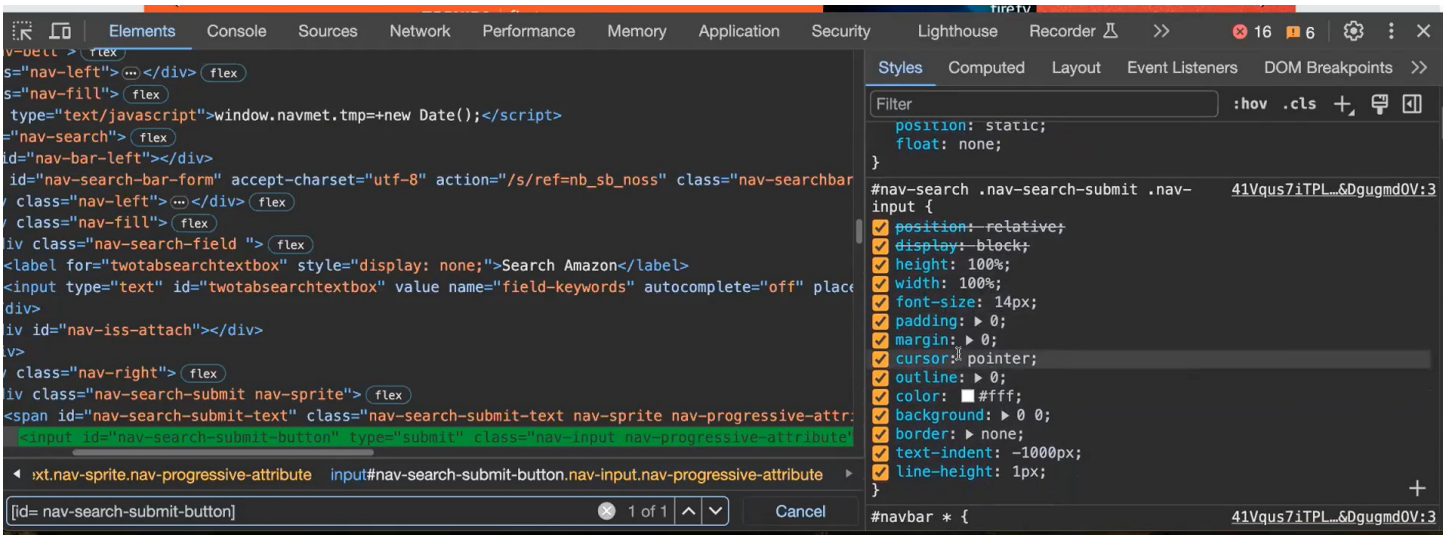
Get CSS VALUE

- **getCssValue()** → gets the value of a CSS property of an web element.
- We have to specify what property we want to get. E.g:

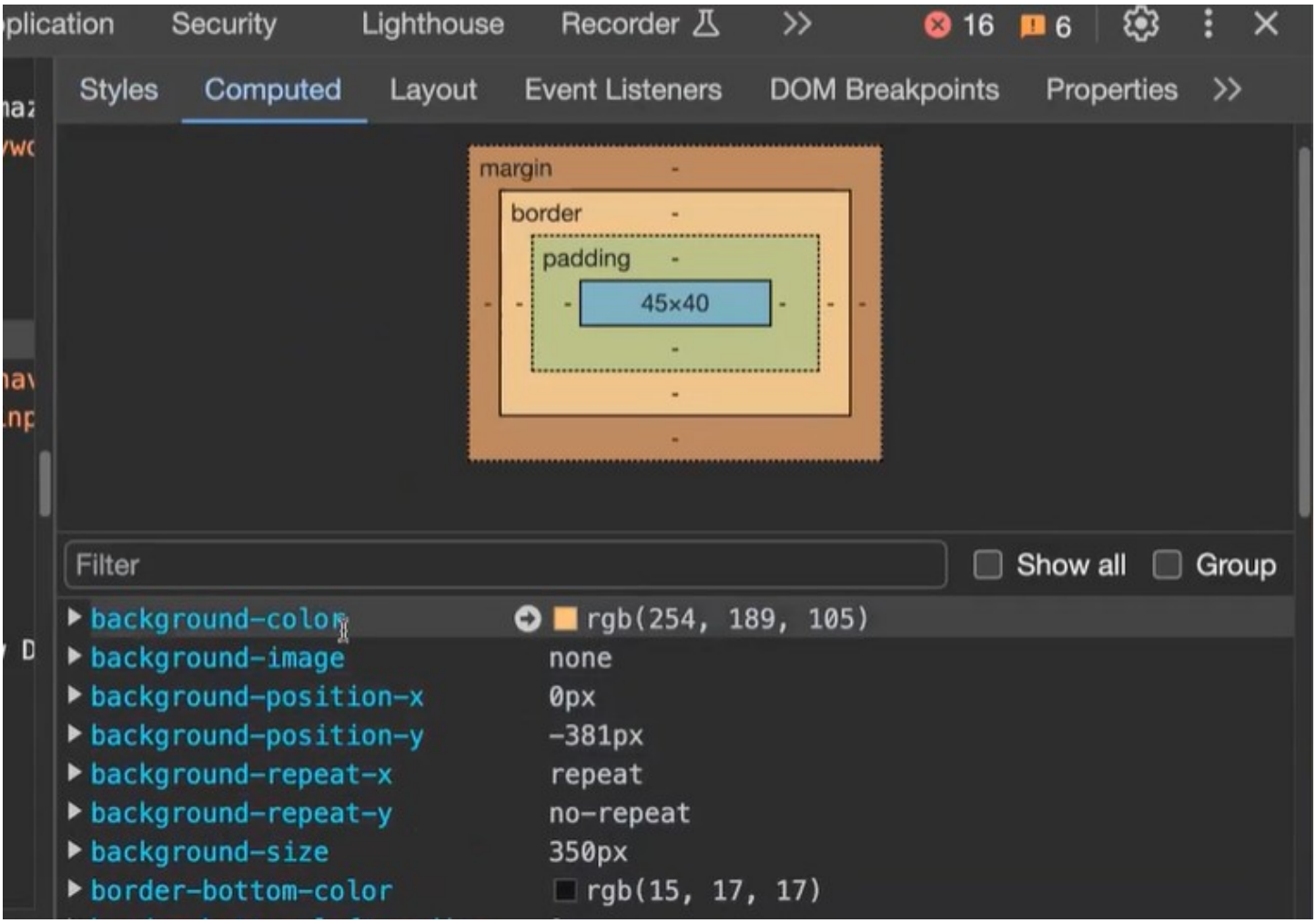
```
.getCssValue("background-color");
.getCssValue("border-bottom-color");
getCssValue("border-bottom-width")
```



Example from Amazon Website:



We take the attibute from : Styles/Computed:



Code:

- Get backgroundColor:

```
// getCssValue() returns a string
        String backgroundColor = searchButton.getCssValue ( "background-color");
```

```
System.out.println("The color of the search button is: " + backgroundColor);
```

```
The color of the search button is: rgba(254, 189, 105, 1)

Process finished with exit code 0
```

There are 3 representation for `background-color:`

```
.header-color {
      background-color: blue;
      background-color: #0101ff;
      background-color: rgba(123,34,34,0);
}
```

- Get Font:

```
//Get the Font
      String font = searchInputBox.getCssValue ( "font-family");
      System.out.println("The font family is: " + font);
```

```
The font family is: Arial, sans-serif

Process finished with exit code 0
```

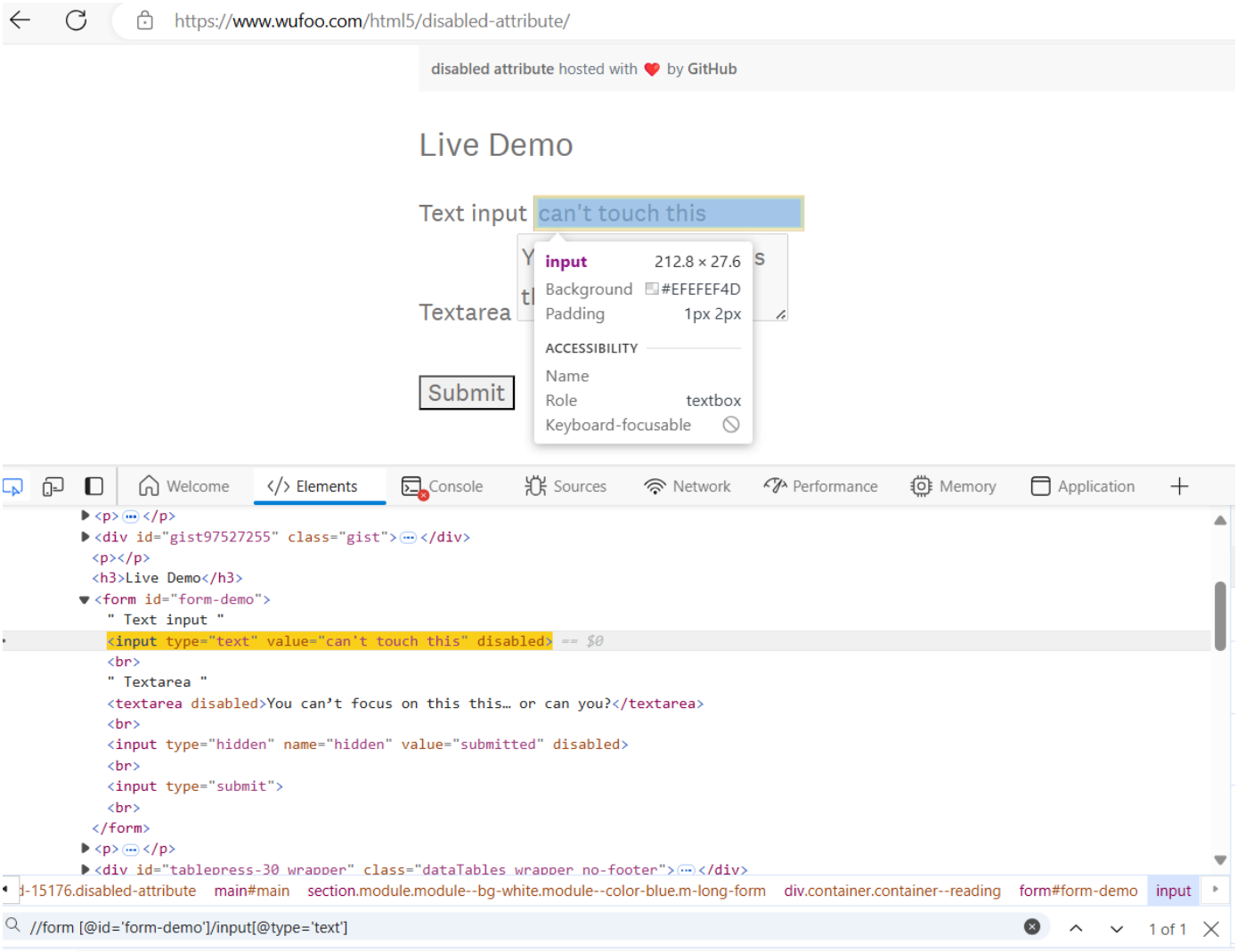
Practice A:

**Navigate to https://www.wufoo.com/html5/disabled-attribute/?
Locate text input field and print on console that it has 'disabled' attribute Locate textarea field and print on console that it has 'disabled' attribute**

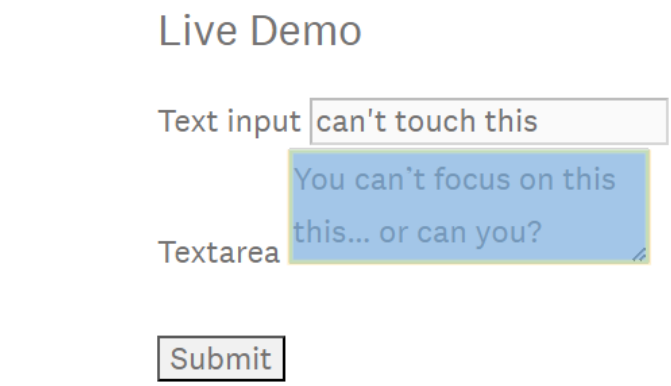
HTML5 Web Resource and Guide | Wufoo

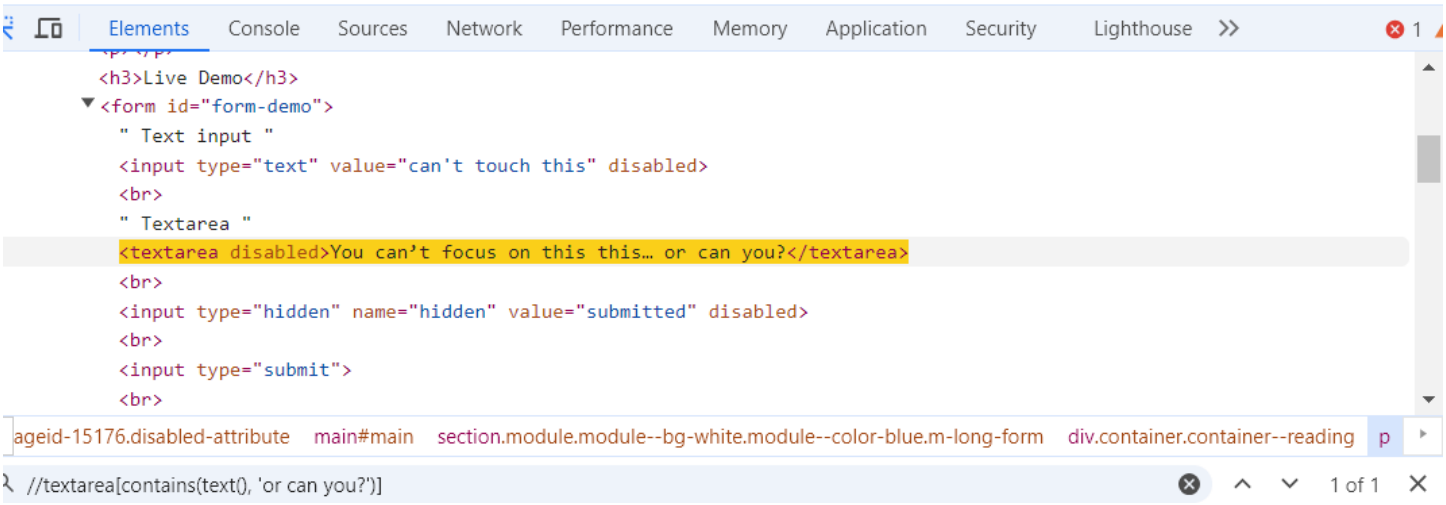
HTML5 Forms: Disabled Type Attribute | Wufoo

//form[@id='form-demo']/input[@type='text']"));



//textarea[contains(text(), 'or can you?')]






```java
package module2;
/*
Navigate to https://www.wufoo.com/html5/disabled-attribute/?
Locate text input field and print on console that it has 'disabled' attribute
Locate textarea field and print on console that it has 'disabled' attribute
**/
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class PracticeClass7 {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        // Setup
        driver.manage().window().maximize();
```

```
        driver.get("https://www.wufoo.com/html5/disabled-attribute/?");

        // Locate text input field and check if it has 'disabled' attribute
        WebElement textInputField = driver.findElement(By.xpath("//form[@id='form-
demo']/input[@type='text']"));
        boolean textInputFieldHasDisabled = !textInputField.isEnabled();
        System.out.println("Text Input Field Has Disabled Attribute: " +
textInputFieldHasDisabled);


        boolean textAreafieldHasDisabled = driver.findElement(By.xpath(
"//textarea[contains(text(), 'or can you?')]")).isEnabled();
        System.out.println("Text Input Field Has Disabled Attribute:" + (!
textInputFieldHasDisabled));
        System.out.println("Text Area Field Has Disabled Attribute:" + (!
textAreafieldHasDisabled));

        driver.quit();
    }
}
```

```
Text Input Field Has Disabled Attribute:false
Text Area Field Has Disabled Attribute:true


Process finished with exit code 0
```

HOW TO USE ChromeDriver / FirefoxDriver / SafariDriver ?

```
package module2;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox. FirefoxDriver;
import org.openqa.selenium.safari.SafariDriver;

import java.sql.SQLException;

import static java.sql.DriverManager.getDriver;

    public class validateSubmitForm {
    public static void main(String[] args) throws SQLException {
        /* Set Up the environment */
// Upcasting. Why?
//WebDriver driver = new ChromeDriver();
        WebDriver driver = getDriver("CHROME");
// test case
// driver.get()
// driver.find
```

```
// driver.click()

    }
        public static WebDriver getDriver(String browsertype) {
            switch (browsertype) {
            case "CHROME":
                return new ChromeDriver();
            break;
            case "FIREFOX":
                return new FirefoxDriver();
            break;
            case "SAFARI":
            return new SafariDriver();
            break;
                default:return new ChromeDriver();
            }
    }}
```

This code is not working !

// In the real environment we will utilize a method that will return a need type of browser

// In such cases we can utilize the concept of Upcasting and Polymorphism...

!!!Missing Part !!!

Steps of Using Interface for : ChromeDriver / FirefoxDriver / SafariDriver

Step 1: Define the WebDriver interface

Create an interface that declares the methods you want to use for browser interactions:

Step 2: Implement the interface for different browsers

Create classes that implement the `BrowserDriver` interface for Chrome, Firefox, and Safari:

Step 3: Use the interface in your main class

Now, you can use the `BrowserDriver` interface in your main class to interact with different browsers without changing your code:

For our example, we will use three files :

ValidateSubmitForm.java

MyClass.java

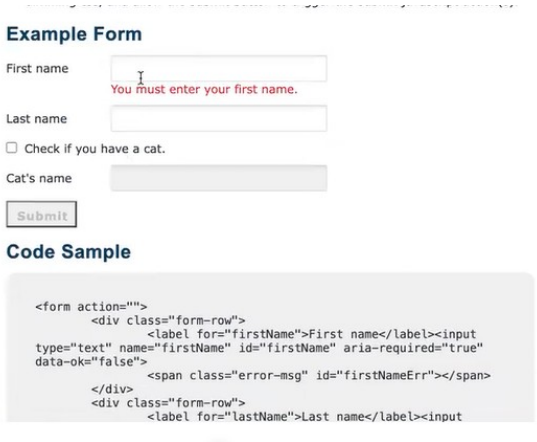
MyInterface.java

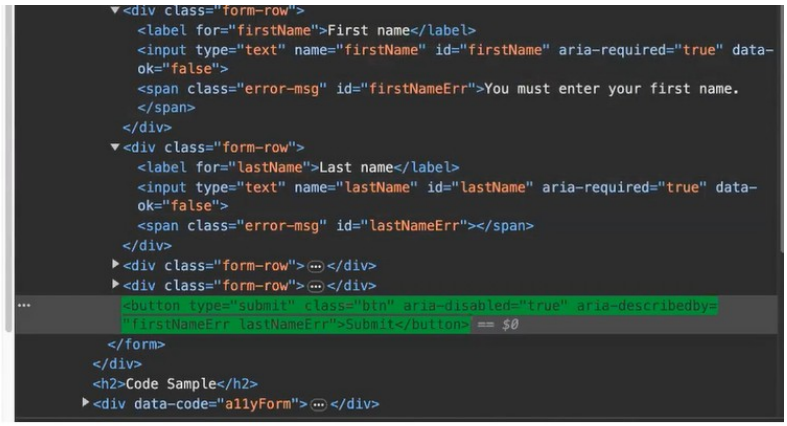
The website:

[Submit Button Disabling (haltersweb.github.io)](https://haltersweb.github.io/Accessibility/submit-disabling.html)

After locating First name, Last name, Submit button (Disabled), we need to click on any place on the page to make Submit button enabled.

We choose to click on (h2) of Example Form. Therefore, we need to locate it by using XPATH.



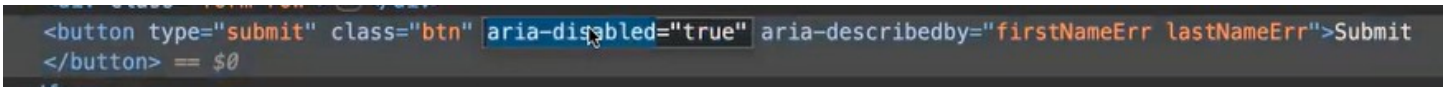


```
package module2;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

import java.sql.SQLException;

import static java.sql.DriverManager.getDriver;

public class validateSubmitForm {
      public static void main(String[] args) throws SQLException {
            /* Set Up the environment */
            WebDriver driver = new ChromeDriver(); //
driver.manage().window().maximize();
            /* navigate to the web-site */
            driver.get("https://haltersweb.github.io/Accessibility/submit-
disabling.html");
            /* Locate the web elements we need to interact with */
            WebElement fNameInputBox = driver.findElement(By.id("firstName"));
            WebElement NameInputBox = driver.findElement(By.id("lastName"));
            WebElement submitButton = driver.findElement(By.cssSelector(".btn"));
            WebElement formHeader = driver.findElement(By.xpath("//h2[text() = 'Example
Form']"));
            /* Create actions with elements */
```

```
        boolean isButtonEnabled = submitButton.isEnabled();
        System.out.println("The button is enabled: " + isButtonEnabled);
```

How to check that the Submit Button is enabled or disabled ?

This code is not working good ! we need to change the boolean to a string by using getAttribute() of aria-disabled for Submit Button to check if it is enabled or disabled:




```
    }
}
```

The result is wrong:

```
The button is enabled: true

Process finished with exit code 0
```

we expected :

The button is enabled:false

```
/* Create actions with elements USING STRING)*/
        String isButtonDisabled = submitButton.getAttribute(  "aria-disabled");
        System.out.println("The button is disabled: " + isButtonDisabled);
```

Result:

```
The button is disabled: true

Process finished with exit code 0
```

If we need to use the boolean, we need to use

```
/* Create actions with elements With Boolean USING PARSEBOOLEAN*/
        boolean isButtonDisabled = Boolean.parseBoolean (submitButton.getAttribute("aria-disabled"));
        System.out.println("The button is disabled: " + isButtonDisabled);
```

Result:

```
The button is disabled: true

Process finished with exit code 0
```

```
package module2;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

import java.sql.SQLException;

import static java.sql.DriverManager.getDriver;

public class validateSubmitForm {
      public static void main(String[] args) throws SQLException, InterruptedException {
            /* Set Up the environment */
            WebDriver driver = new ChromeDriver(); //
driver.manage().window().maximize();
            /* navigate to the web-site */
            driver.get("https://haltersweb.github.io/Accessibility/submit-
disabling.html");
            /* Locate the web elements we need to interact with */
            WebElement fNameInputBox = driver.findElement(By.id("firstName"));
            WebElement LNameInputBox = driver.findElement(By.id("lastName"));
            WebElement submitButton = driver.findElement(By.cssSelector(".btn"));
            WebElement formHeader = driver.findElement(By.xpath("//h2[text() = 'Example
Form']"));
            /*
             * Create actions with elements (WRONG RESULT WITH BOOLEAN. THEREFORE, WE USE
             * STRING)
             */
            // boolean isButtonEnabled = submitButton.isEnabled();
            // System.out.println("The button is enabled: " + isButtonEnabled);

            /* Create actions with elements USING STRING) */
            // String isButtonDisabled = submitButton.getAttribute( "aria-disabled");
            // System.out.println("The button is disabled: " + isButtonDisabled);
            /* Create actions with elements With Boolean USING PARSEBOOLEAN */
            // boolean isButtonDisabled = Boolean.parseBoolean
            // (submitButton.getAttribute("aria-disabled"));
            // System.out.println("The button is disabled: " + isButtonDisabled);

            /* Create actions with elements */
```

```
            boolean isButtonDisabled =
Boolean.parseBoolean(submitButton.getAttribute("aria-disabled"));
            System.out.println("The button is disabled: " + isButtonDisabled);

            /* Checking the submit button color when is still disabled */
            String submitButtonColor = submitButton.getCssValue("color");
            System.out.println("Current text color of submit button: " +
submitButtonColor);
            Thread.sleep(2000);
            /* Fill out the form to enable the submit button */
            fNameInputBox.sendKeys("Omar");
            Thread.sleep(2000);
            LNameInputBox.sendKeys("Ramo");
            Thread.sleep(2000);
            boolean isButtonDisabledAfterFill =
Boolean.parseBoolean(submitButton.getAttribute("aria-disabled"));
            System.out.println("The button is disabled after filling the form: " +
isButtonDisabledAfterFill);
            /* Click any element on the page */
            formHeader.click();
            Thread.sleep(2000);
            boolean isButtonDisabledAfterHeaderClick =
Boolean.parseBoolean(submitButton.getAttribute("aria-disabled"));
            System.out.println("The button is disabled after clicking the header: " +
isButtonDisabledAfterHeaderClick);

            /* Checking the submit button color if it is still enabled */
            submitButtonColor = submitButton.getCssValue("color");
            System.out.println("Current text color of submit button: " +
submitButtonColor);
            driver.quit();
      }
}
```

```
The button is disabled: true
Current text color of submit button: rgba(187, 187, 187, 1)
The button is disabled after filling the form: true
The button is disabled after clicking the header: false
Current text color of submit button: rgba(22, 30, 37, 1)

Process finished with exit code 0
```

SendKeys has many value to use !

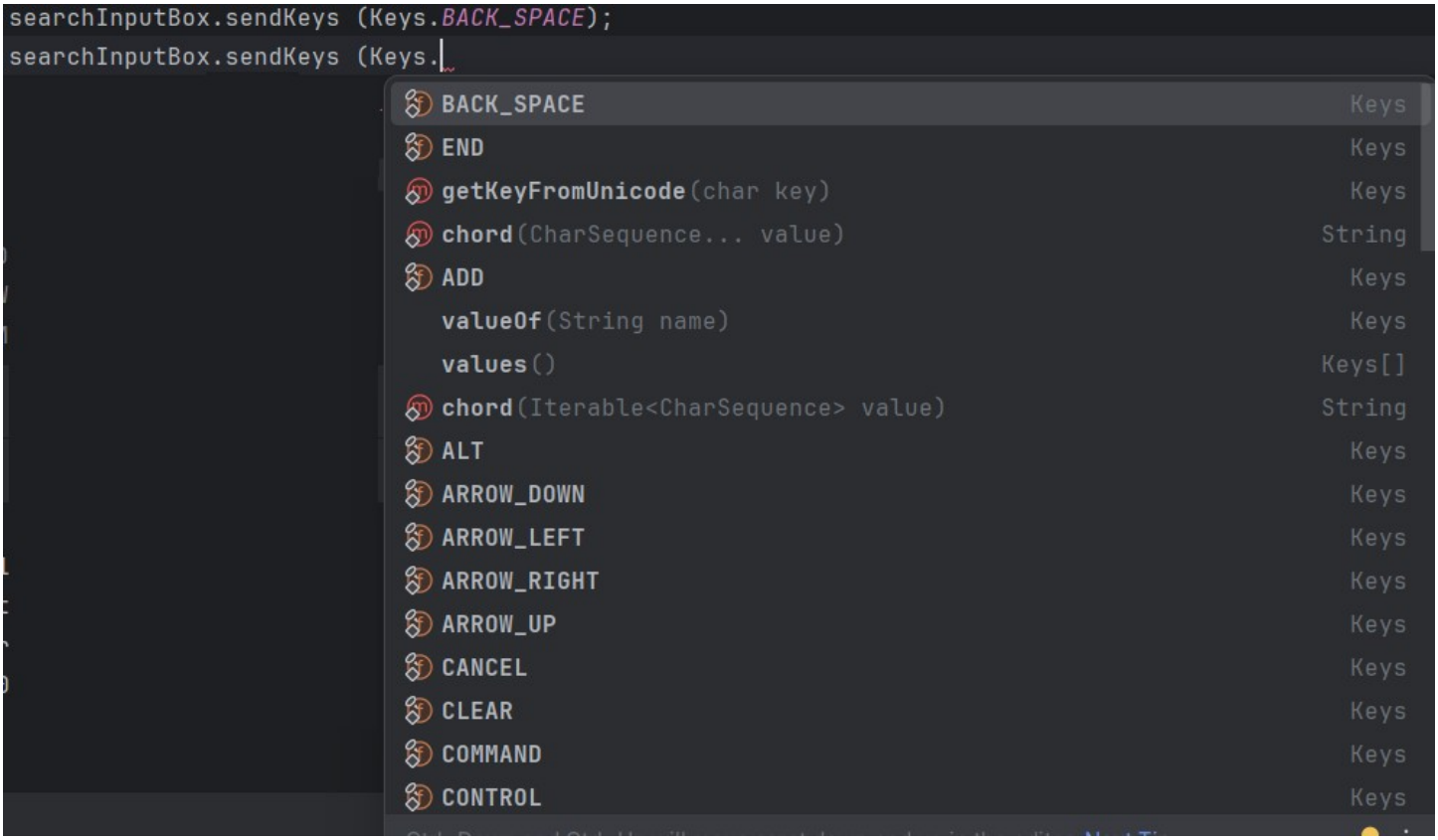
Multiple Keys with Keys.chord in SendKeys method

- We can send multiple keys using sendkeys() in webdriver.
- For example if we want press shift button and enter some text to a field.

```
driver.findElement(By.xpath("//input[@id='textbar']")).sendKeys(Keys.chord(Keys.SHIFT, "userName"));
```

- You can also use it to simulate CTRL+A
  - `driver.findElement(By.id("fta")).sendKeys(Keys.chord(Keys.CTRL, "a"));`

Note: There are alternative ways of achieving these tasks, so these won't really be used much.



Task using sendkeys + other commands:

*** Go to Amazon In search bar enter iPhone 16 Remove the last character Insert**

*** "5 Pro Max" Click Enter/Return Highlight all text that we entered Win: ctrl +**

```
     * Mac: Command + "a" Move to the left with Delete one
character Move arrow 3
     * times down Then 1 time up Click Enter/Return
     */


package module3;

import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class KeysPractice {

     public static void main(String[] args) throws InterruptedException {
// Set up
          WebDriver driver = new ChromeDriver();
          driver.manage().window().maximize();
          /* Navigate to amazon web app */
          driver.get("https://www.amazon.com/");
          /* Locate the search input box */
          WebElement searchInputBox =
driver.findElement(By.id("twotabsearchtextbox"));
          Thread.sleep(2000);
          /* In search bar enter iPhone 16 */
          searchInputBox.sendKeys("iPhone 16");
          Thread.sleep(2000);
          /* Remove the last character */
          searchInputBox.sendKeys(Keys.BACK_SPACE);
          Thread.sleep(2000);
          /* Insert "5 Pro Max" */
          searchInputBox.sendKeys("5 Pro Max");
          Thread.sleep(2000);

          // Dif ENTER and RETURN
          // Windows users - ENTER
          // MAC users - RETURN
          // Also for all OS the keypad has ENTER as well
          searchInputBox.sendKeys(Keys.ENTER);
          Thread.sleep(2000);
          /* Remove the last character */
          searchInputBox.sendKeys(Keys.BACK_SPACE);
          Thread.sleep(2000);
          /* Insert "5 Pro Max" */
          searchInputBox.sendKeys("5 Pro Max");
          Thread.sleep(2000);
          /* Highlight all text */
// Win users:
// searchInputBox.sendKeys(Keys.CONTROL);
// Mac users:
```

```
            searchInputBox.sendKeys(Keys.COMMAND + "a");
            Thread.sleep(2000);
            searchInputBox.sendKeys(Keys.ARROW_LEFT);
            Thread.sleep(2000);
            /* Delete one character */
            searchInputBox.sendKeys(Keys.DELETE);
            Thread.sleep(2000);
            /* Move down 3 times */
            searchInputBox.sendKeys(Keys.ARROW_DOWN);
            Thread.sleep(1000);
            searchInputBox.sendKeys(Keys.ARROW_DOWN);
            Thread.sleep(1000);
            searchInputBox.sendKeys(Keys.ARROW_DOWN);
            Thread.sleep(1000);
            searchInputBox.sendKeys(Keys.ARROW_UP);
            Thread.sleep(1000);
            driver.quit();
      }
}
```

Practice:

- Navigate to http://demo.automationtesting.in/Register.html
- Locate the element for 'firstname'
- Pass your first name using sendkeys method
- Move to 'lastname' field using Keys.TAB and enter the last name
- Note: You can use Thread.sleep()

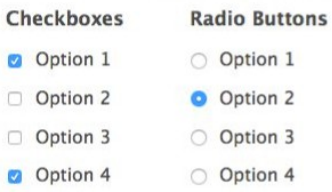
Register (automationtesting.in)

```
package module3;

import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class Practice1 {
      public static void main(String[] args) throws InterruptedException {

            // Set up
            WebDriver driver = new ChromeDriver();
            driver.manage().window().maximize();
            /* Navigate to amazon web app */
            driver.get("https://demo.automationtesting.in/Register.html");
```

```
            WebElement firstNameInputBox =
driver.findElement(By.cssSelector("[placeholder = 'First Name']"));
            firstNameInputBox.sendKeys("Omar");
            Thread.sleep(3000);
            /* FIRST METHOD: Navigate to next web element on the page using TAB */
            firstNameInputBox.sendKeys(Keys.TAB + "Ramo");
            Thread.sleep(3000);

            // SECOND METHOD: Using KEYS.CHORD TO SEND MORE THAN ONE COMMAND AT
THE SAME
            // TIME
            // When you have multiple Keys to send we can use Keys.chord
(Keys.Key, Keys.
            // Key, Keys. Key, Keys.Key);
            firstNameInputBox.sendKeys(Keys.chord(Keys.TAB, "Ramo", Keys.TAB, "My
Address is Incognito", Keys.TAB,
                        "email@email.com", Keys.TAB, "1233211221"));
            Thread.sleep(3000);
            driver.quit();
      }
}
```

Checkboxes, Radio Buttons

- We can use **click()** method to check the checkbox and uncheck the checkbox
  - **isSelected()** method is used to know whether the **Checkbox** is toggled on or off. It is also **used on radio buttons and dropdowns** in selenium.
- We can use **click()** method to click the radio button, as we know we cannot unselect a radio button by clicking it.
  - To unselect a radio button we need to click another radio button.

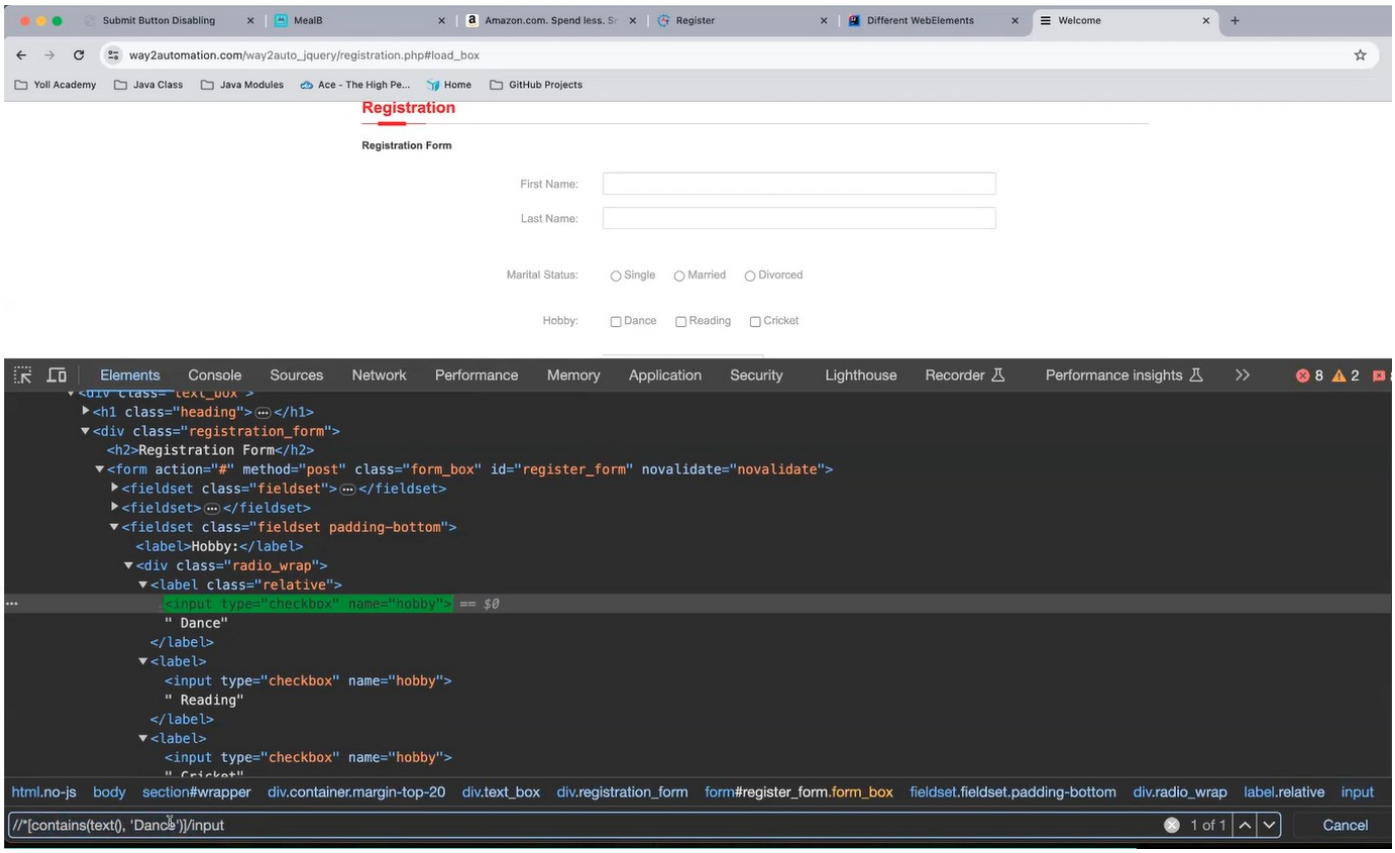


Task 1

- Navigate to http://demo.automationtesting.in/Register.html
- Fill Out the form elements except Languages, Skills, Country, Select Country, date of birth
- Click the submit button

[Register (automationtesting.in)]

Task 2



Check the checkbox 1 and 2 : Dance /Reading.

Uncheck the checkbox 1.

Verify checkbox 1 and 2 are checked or not

Method 1:

```
package module3;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class CheckBoxAndRadioButton{
public static void main(String[] args) throws InterruptedException {

    WebDriver driver = new ChromeDriver();

driver.get("https://www.way2automation.com/way2auto_jquery/registration.php#load_box");
```

```
    Thread.sleep( 2000);
    /* How to select a single checkbox */
    WebElement checkBox1 = driver.findElement(By.xpath(   "//*[contains(text(),
'Dance')]/input"));
    checkBox1.click();
    Thread.sleep( 2000);
    /* To uncheck a checkbox we simply click on it again */
    checkBox1.click();

    WebElement checkBox2 = driver.findElement(By.xpath(   "//*[contains(text(),
'Reading')]/input"));
    checkBox1.click();
    Thread.sleep( 2000);
    // isSelected() method will return a boolean if a element is checked or chosen
    // Lets print the checkbox1 and checkbox2 state
    System.out.println("Checkbox 1 is selected: " + checkBox1.isSelected());
    System.out.println("Checkbox 2 is selected: " + checkBox2.isSelected());

    driver.quit();
}}
```

```
Checkbox 1 is selected: true
Checkbox 2 is selected: false


Process finished with exit code 0
```

Method 2:

Check the checkbox 1 and 2 and 3 : Dance /Reading / Cricket.

Check the Radio Buttons: Single, Married, and Divorced.

Verify the Checkboxes and Radio buttons are selected or not.

Use the method selectCheckbox() that contains two parameters (Driver, checkboxName)**to avoid repetition of the** checkboxName **XPATH for each element when Checking or unchecking the checkboxes by making checkboxName more dynamic**

```
 String checkboxXpath = "//*[contains(text(),'"+checkboxName+"')]/input";
"//*[contains(text(), 'Dance')]/input"));
"//*[contains(text(), '')]/input"));
"//*[contains(text(), '" "')]/input"));
" " = For String
"//*[contains(text(), '"checkboxName"')]/input"));
```

Calling the method:

```
  /* How to select a single checkbox */
    selectCheckbox (driver,  "Dance");
    Thread.sleep( 2000);
    /* To uncheck a checkbox we simply click on it again */
    selectCheckbox (driver,  "Dance");
```

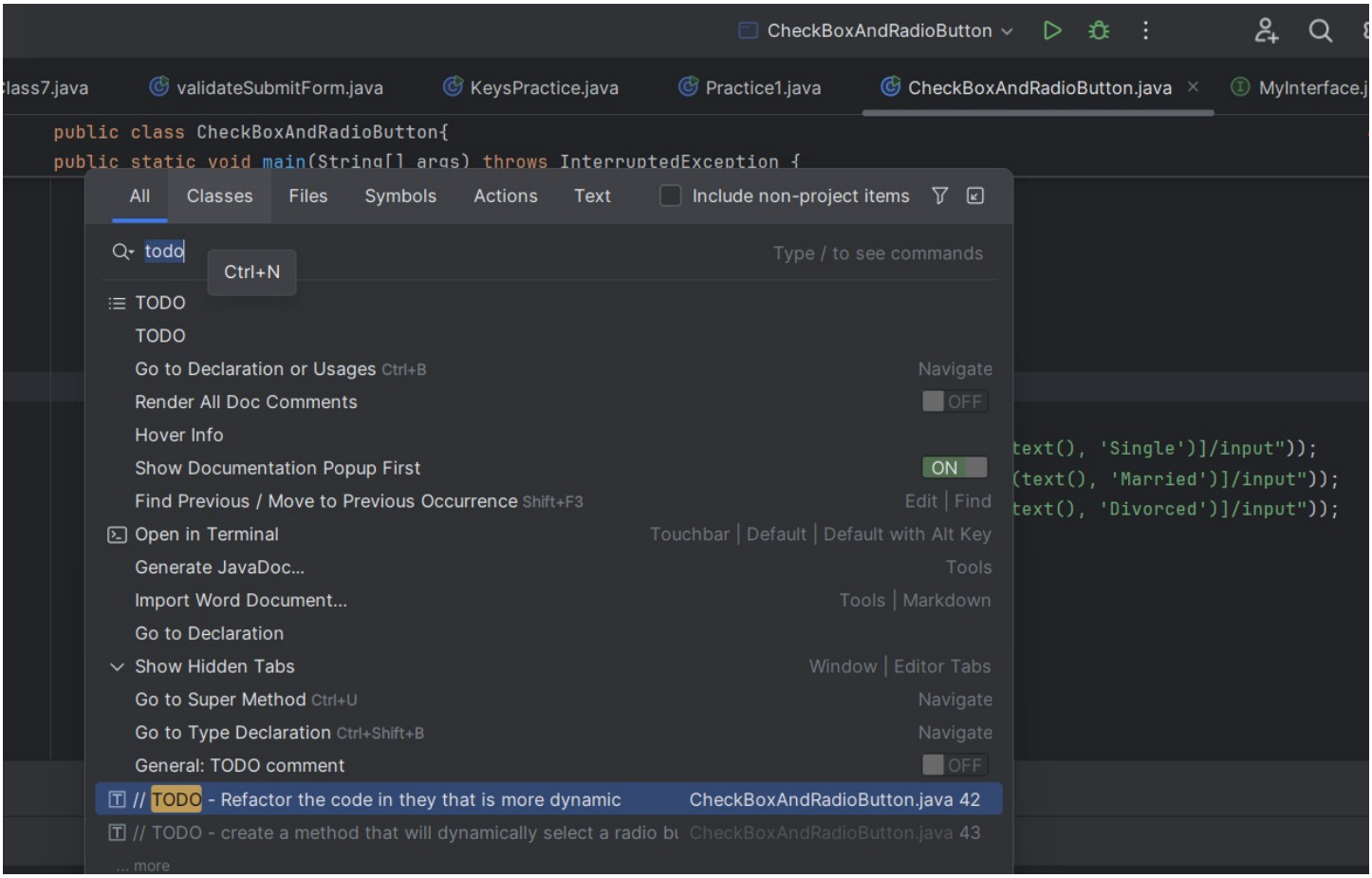
```
    selectCheckbox (driver,  "Reading");


package module3;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class CheckBoxAndRadioButton{
public static void main(String[] args) throws InterruptedException {

    WebDriver driver = new ChromeDriver();

driver.get("https://www.way2automation.com/way2auto_jquery/registration.php#load_box");
    Thread.sleep( 2000);
    /* How to select a single checkbox */
    selectCheckbox (driver,  "Dance");
    Thread.sleep( 2000);
    /* To uncheck a checkbox we simply click on it again */
    selectCheckbox (driver,  "Dance");
    selectCheckbox (driver,  "Reading");
    Thread.sleep( 2000);

    // isSelected() method will return a boolean if a element is checked or chosen
    // Lets print the checkbox1 and checkbox2 state
    // This part is done specifically for debugging
    WebElement checkBox1 = driver.findElement(By.xpath(  "//*[contains(text(),
'Dance')]/input"));
    WebElement checkBox2 = driver.findElement(By.xpath(  "//*[contains(text(),
'Reading')]/input"));
    System.out.println("Checkbox 1 is selected: " + checkBox1.isSelected());
    System.out.println("Checkbox 2 is selected:" + checkBox2.isSelected());
// Radiobutton
    WebElement radioButton1 = driver.findElement(By.xpath(  "//*[contains(text(),
'Single')]/input"));
    WebElement radioButton2 = driver.findElement(By.xpath(  "//*[contains(text(),
'Married')]/input"));
    WebElement radioButton3 = driver.findElement(By.xpath( "//*[contains(text(),
'Divorced')]/input"));
    radioButton1.click();
    radioButton2.click();
    radioButton3.click();

    driver.quit();
}

    /* Create a method that will select a checkbox by text */
    public static void selectCheckbox (WebDriver driver, String checkboxName) {

       String checkboxXpath = "//*[contains(text(),'"+checkboxName+"')]/input";

        driver.findElement(By.xpath(checkboxXpath)).click();

    }
}
```

TODO – Refactor

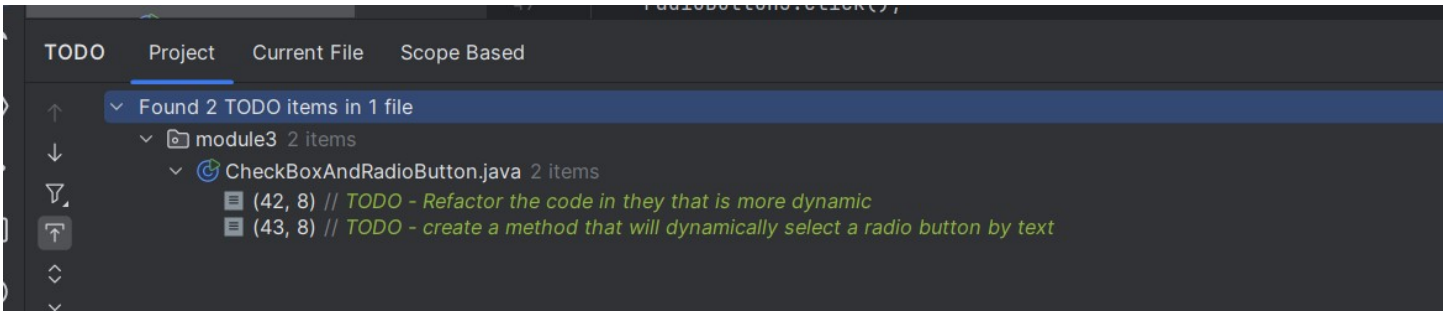
Refactoring the code involves improving its structure and readability while preserving its functionality.

```
// TODO - Refactor the code in they that is more dynamic
// TODO - create a method that will dynamically select a radio button by text
```

On **Intellij**, Go to the **serach button**, and tape with capital letters: **TODO**



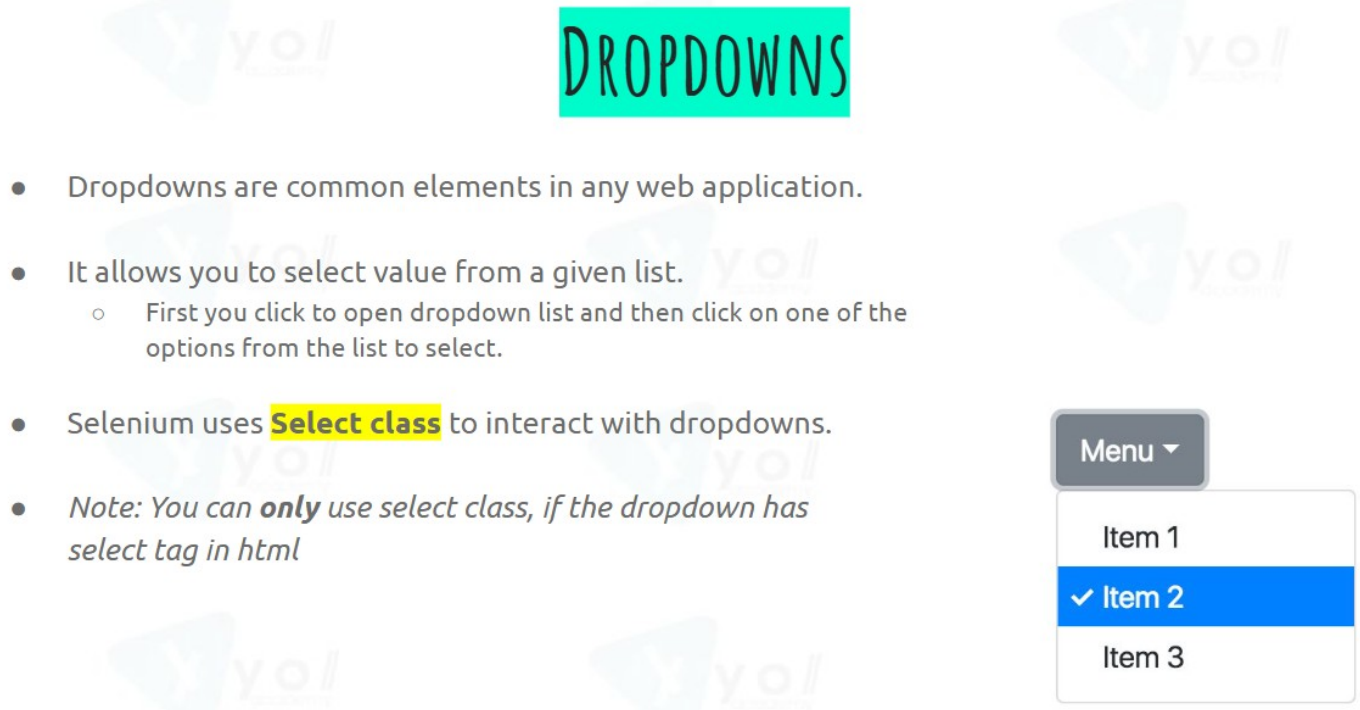
You can see the whole TODO that the project has.


TODO
Project
Current File
Scope Based
Found 2 TODO items in 1 file
module3 2 items
CheckBoxAndRadioButton.java 2 items
(42, 8) // TODO - Refactor the code in they that is more dynamic
(43, 8) // TODO - create a method that will dynamically select a radio button by text

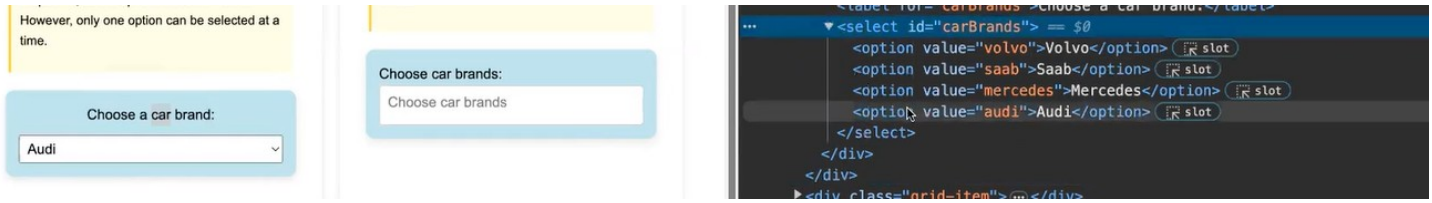
Question: When do you work with TODO ?

Answer: Sometimes, you have a lot work to do, sometimes not. And we can say to the manager that I refactoring some codes, thechnical debts when I do not have any **ticket**.

Dropdowns

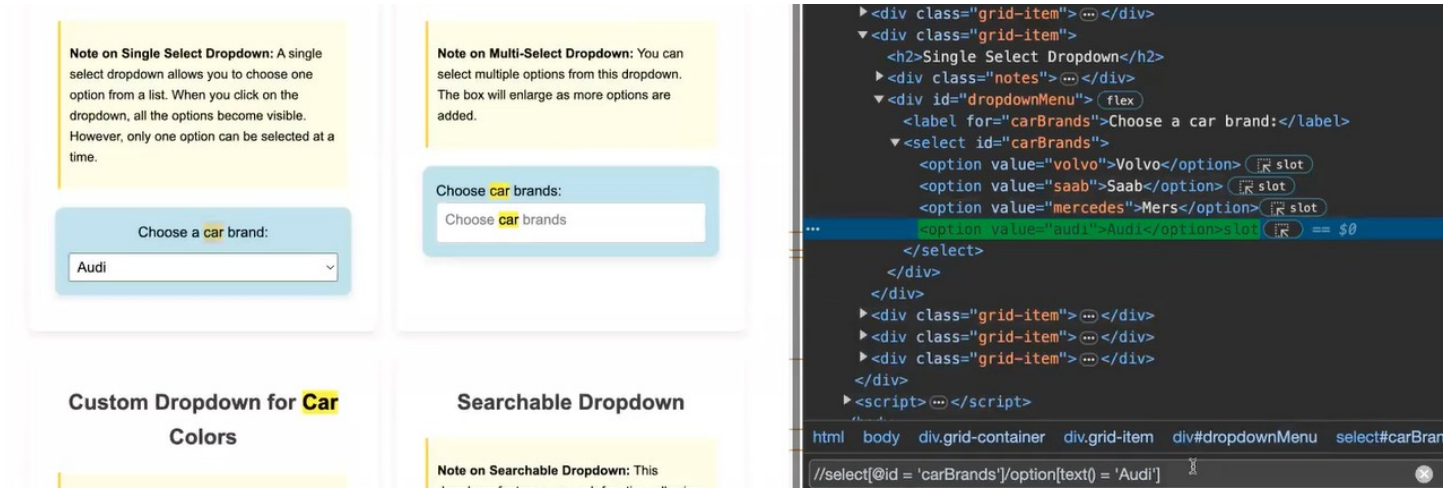

DROPDOWNS
Dropdowns are common elements in any web application.
It allows you to select value from a given list.
First you click to open dropdown list and then click on one of the options from the list to select.
Selenium uses Select class to interact with dropdowns.
Note: You can only use select class, if the dropdown has select tag in html
Menu
Item 1
Item 2
Item 3

We use Select class because when you inspect the dropdrown, will you **SELECT TAG** , and it has options children:



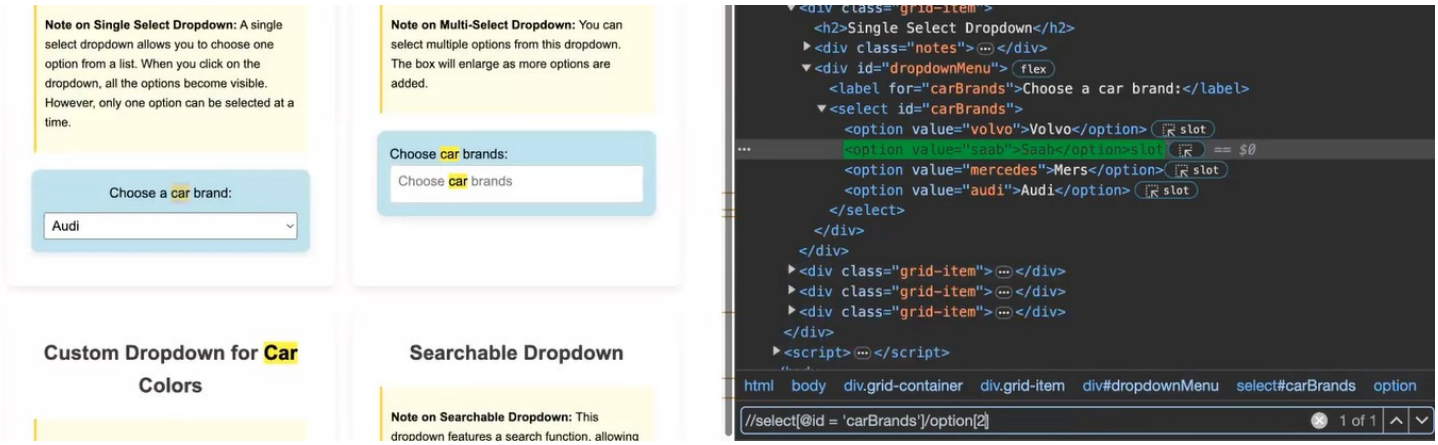
XPATH of AUDI:

//select[@id=’carBRANDS’]/option[text()=’Audi’]



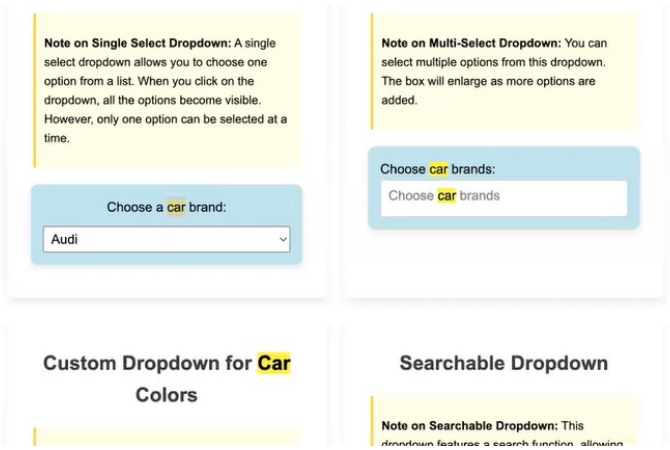
XPATH of SAAB CAR USING INDEX:

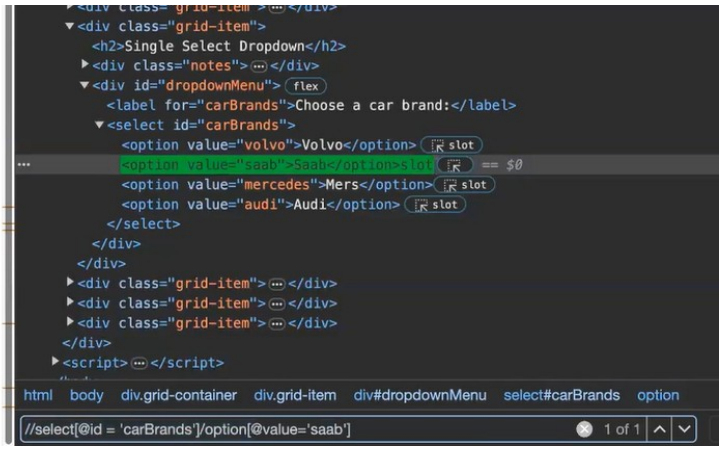
//select[@id=’carBRANDS’]/option(2)



XPATH of SAAB CAR USING value attribute:

//select[@id=’carBRANDS’]/option[@value=’saab’]


Note on Single Select Dropdown: A single select dropdown allows you to choose one option from a list. When you click on the dropdown, all the options become visible. However, only one option can be selected at a time.
Choose a car brand:
Audi
Note on Multi-Select Dropdown: You can select multiple options from this dropdown. The box will enlarge as more options are added.
Choose car brands:
Choose car brands
Custom Dropdown for Car Colors
Searchable Dropdown
Note on Searchable Dropdown: This



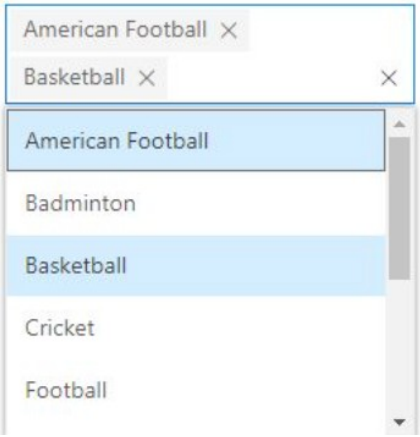
TYPES OF DROPDOWNS

Single Value Dropdowns

- You can select only one value from the dropdown list.
- %90 of dropdowns are single select

Multiple Value Dropdowns

- You can select multiple values from the dropdown list.



Single Select Dropdown

- Dropdown options are indexed with numbers. They start from 0.
- Each option has **value** attribute
- Each option has **visible text**.
  - This is the actual text humans get to see when opened dropdown.
- Note that, values and visible texts may not be the same.

Interact with Dropdown

In order to select a value from dropdown we have to:

1. **Locate** the dropdown element
```
WebElement myDropdown = driver. findElement(By.id("wer"));
```

2. **Create an object from Select class** by passing "dropdown element" as parameter to constructor
```
Select dropdown = new Select(myDropdown );
```

3. Get all the options present in the dropdown.
   a. Note that this returns you elements not their text yet.
```
List <WebElement> allElements = dropdown. getOptions();
```

4. We can now print the options by looping through.
```
for (WebElement element : allElements) {
    System.out.println(element.getText());
}
```

SELECT VALUE FROM DROPDOWN

- **Select the desired option** using **index**.

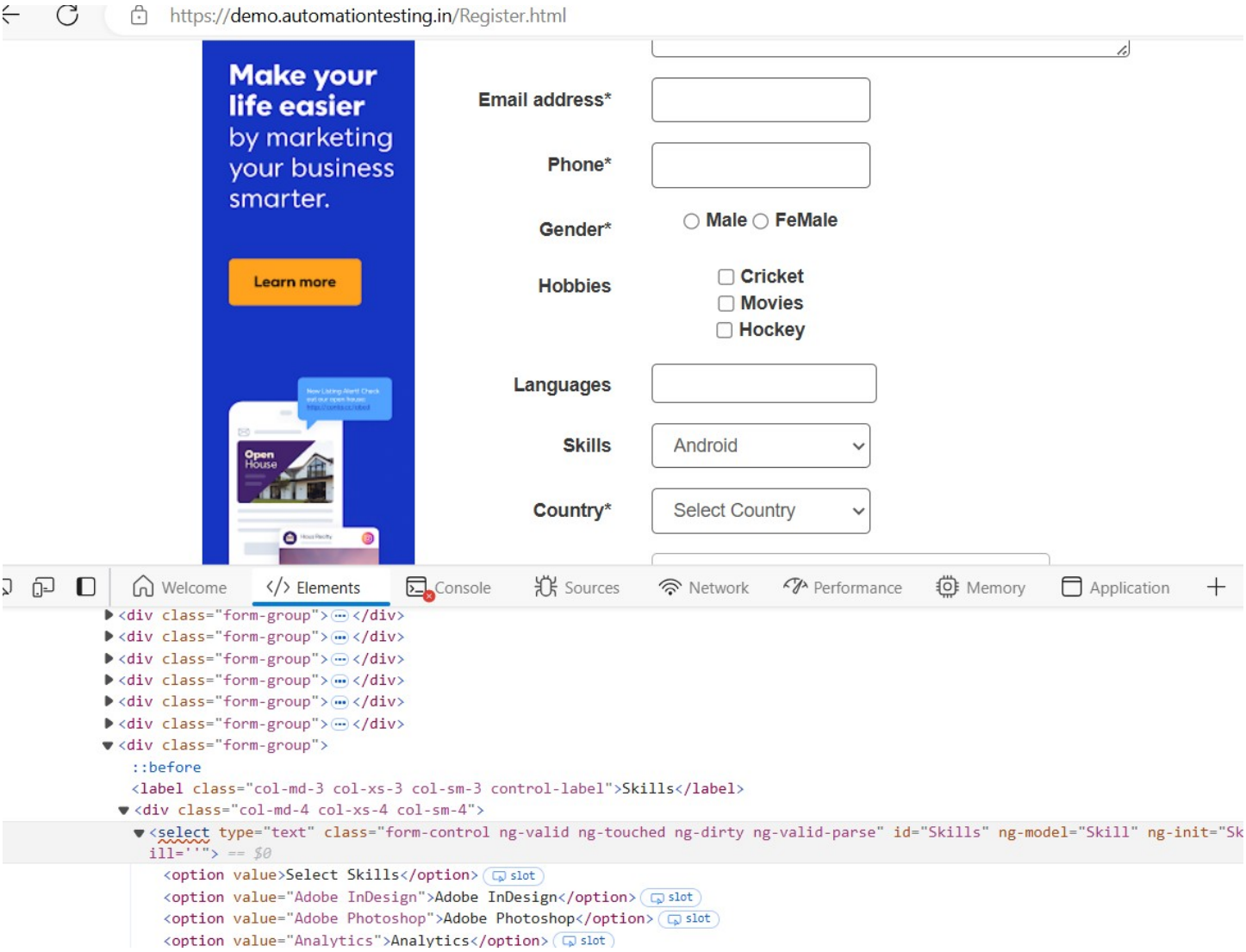
```
dropdown.selectByIndex(0);
```

selectByIndex() is one of the ways to select from the dropdown.

- There are also other ways: **by value** and **by visible text**

```
myDropdown.selectByValue("New York");
```

```
myDropdown.selectByVisibleText("Bing");
```




```
package module3;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
```

```java
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;

public class SelectDropdownPractice {
    public static void main(String[] args) throws InterruptedException {
        WebDriver driver = new ChromeDriver();
        // maximize()
        driver.get("https://demo.automationtesting.in/Register.html");
        /*
         * Working with Select Dropdowns
         * - How do we automate it?
         * - First locate the dropdown
         * - Second, create a class Select by initializing the object with web element
located previously
         **/


// Locate dropdown
        WebElement skillsDropdown = driver.findElement(By.id("Skills"));
// Create a select object
// !!! We can use Select class on the dropdowns that have select tag !!!
        Select dropdown = new Select(skillsDropdown);
        Thread.sleep(1000);
// Select an option by index
// Remember, here index starts from @
        dropdown.selectByIndex (1);
        Thread.sleep(2000);
// Select by visible text
        dropdown.selectByVisibleText("CSS");
        Thread.sleep(3000);
// Select by value attribute dropdown.selectByValue ("HTML"); Thread.sleep(millis: 3000);
        driver.quit();
    }
}
```

TASK

```
Navigate to: https://demo.automationtesting.in/Register.html
Locate the Year dropdown - it is a part of DOB
Use Select class
Choose 1916 option by index
Choose 1956 by value
Choose 2010 by visible text
Finally print all options in the console
```

Find if Option is present

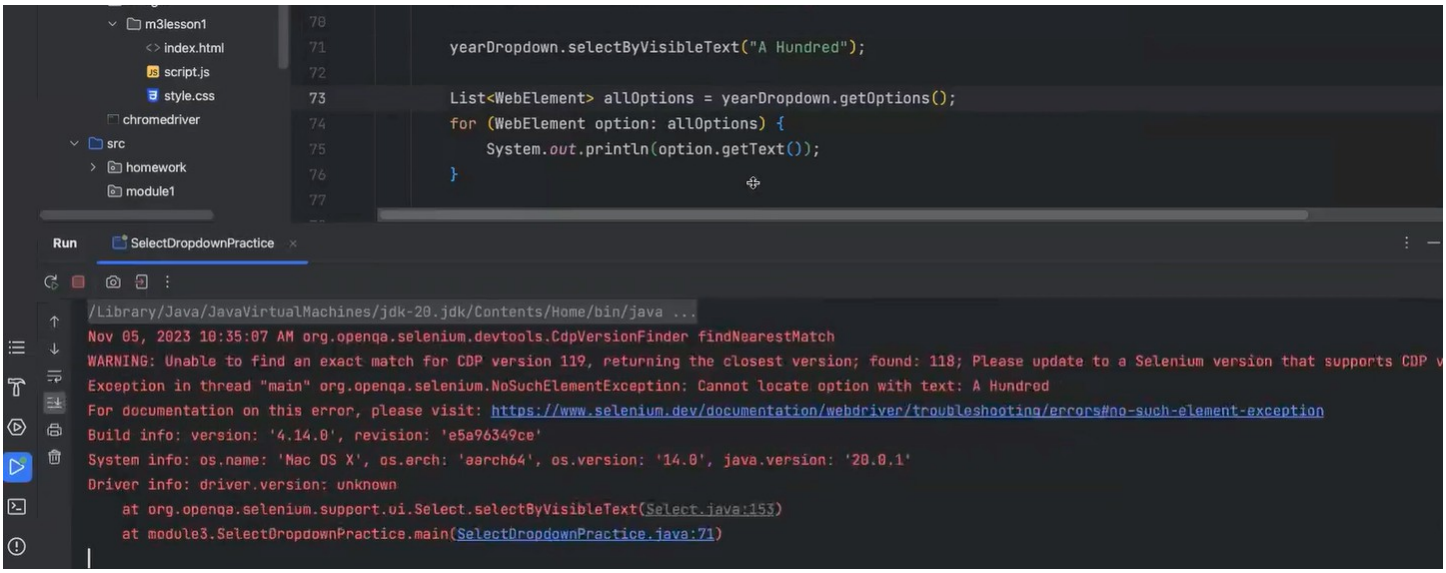
How to find if an element does not exist ?

- We can find if a particular option is present in the dropdown list, using getOptions.
    - ```
      Select dropdown = new Select(dropdownElement);
      List<WebElement> allElements = dropdown.getOptions();
      boolean presenceOfElement = false;
      for (WebElement element : allElements) {
              String dropdownOptionValue = element.getText();
              if (dropdownOptionValue.equals("Baku")) {
                  presenceOfElement = true;
                  }
      }
      ```
- Alternatively, we can use xpath to find it as well:
    - ```
      driver.findElement(By.xpath("//select[@id='editor']//option[text()='Baku']" ));
      ```

If the program does not find the value : "**A Hundred**", it will throw an exception.Therfore, we use **IF CONDITION TO AVOID THE EXCEPTION.**

```
Select yearDropdown = new Select(dropdown);
yearDropdown.selectByIndex(1);
yearDropdown.selectByValue("1956");
yearDropdown.selectByVisibleText("2010");

yearDropdown.selectByVisibleText("A Hundred");

List<WebElement> allOptions = yearDropdown.getOptions();
for (WebElement option: allOptions) {
    System.out.println(option.getText());
}
```

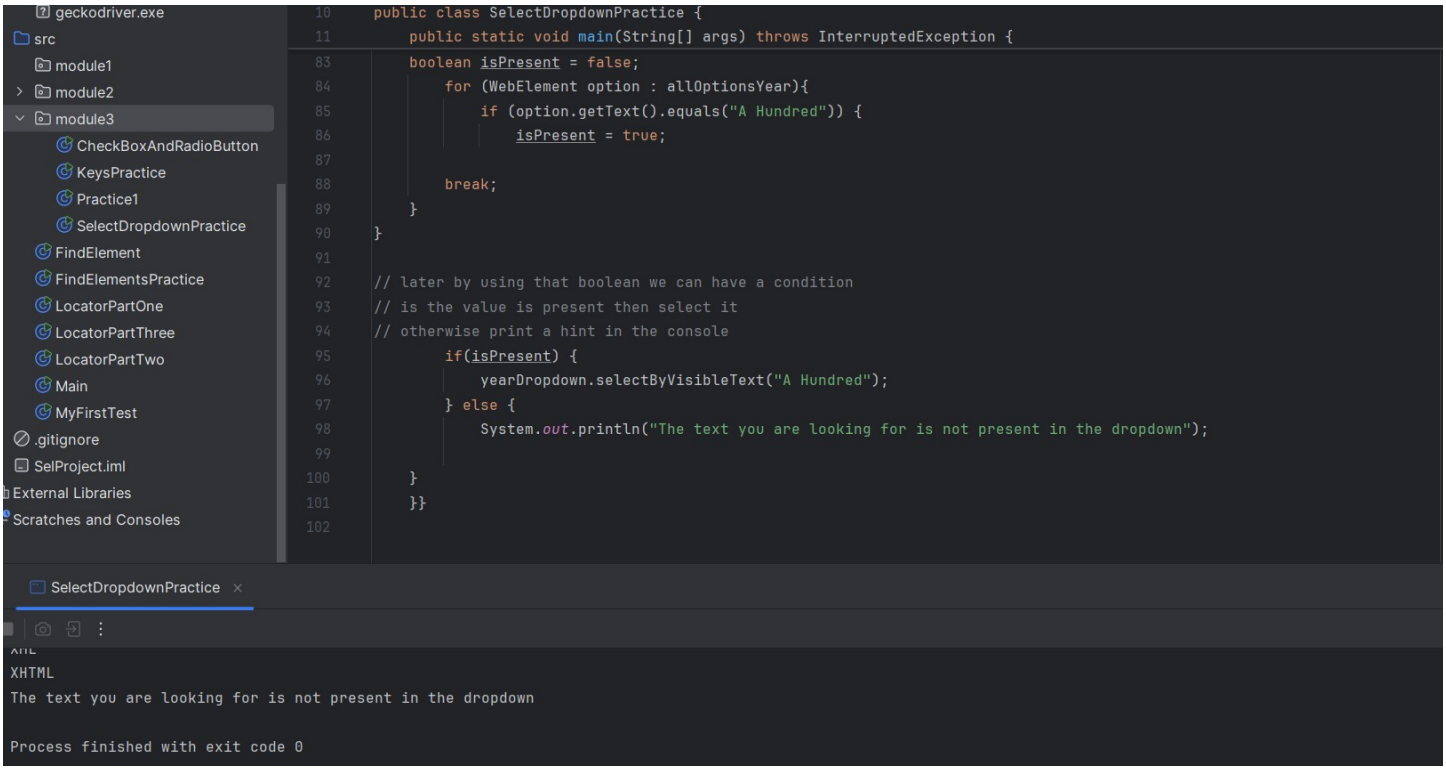
Exception:


m3lesson1
index.html
script.js
style.css
chromedriver
src
homework
module1
yearDropdown.selectByVisibleText("A Hundred");
List<WebElement> allOptions = yearDropdown.getOptions();
for (WebElement option: allOptions) {
System.out.println(option.getText());
}
Run
SelectDropdownPractice
/Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java ...
Nov 05, 2023 10:35:07 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 119, returning the closest version; found: 118; Please update to a Selenium version that supports CDP
Exception in thread "main" org.openqa.selenium.NoSuchElementException: Cannot locate option with text: A Hundred
For documentation on this error, please visit: https://www.selenium.dev/documentation/webdriver/troubleshooting/errors#no-such-element-exception
Build info: version: '4.14.0', revision: 'e5a96349ce'
System info: os.name: 'Mac OS X', os.arch: 'aarch64', os.version: '14.0', java.version: '20.0.1'
Driver info: driver.version: unknown
at org.openqa.selenium.support.ui.Select.selectByVisibleText(Select.java:153)
at module3.SelectDropdownPractice.main(SelectDropdownPractice.java:71)

With IF CONDITION:

Result:

The text you are looking for is not present in the dropdown




```
package module3;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;
```

```java
import java.util.List;

public class SelectDropdownPractice {
	public static void main(String[] args) throws InterruptedException {
		WebDriver driver = new ChromeDriver();
		// maximize()
		driver.get("https://demo.automationtesting.in/Register.html");
		/*
		 * Working with Select Dropdowns - How do we automate it? - First locate the
		 * dropdown - Second, create a class Select by initializing the object with
web
		 * element located previously
		 **/

// Locate dropdown
		WebElement skillsDropdown = driver.findElement(By.id("Skills"));
// Create a select object
// !!! We can use Select class on the dropdowns that have select tag !!!
		Select dropdown = new Select(skillsDropdown);
		Thread.sleep(1000);
// Select an option by index
// Remember, here index starts from @
		dropdown.selectByIndex(1);
		Thread.sleep(2000);
// Select by visible text
		dropdown.selectByVisibleText("CSS");
		Thread.sleep(3000);
// Select by value attribute
		dropdown.selectByValue("HTML");
		Thread.sleep(3000);

		dropdown.selectByValue("Java");
		Thread.sleep(5000);
// Get all options
		List<WebElement> options = dropdown.getOptions();
		for (WebElement option : options) {
			System.out.println(option.getText());

		}
		/*
		 * Navigate to: https://demo.automationtesting.in/Register.html Locate the
Year
		 * dropdown it is a part of DOB Use Select class Choose 1916 option by index
		 * Choose 1956 by value Choose 2010 by visible text Finally print all options
in
		 * the console
		 */

		driver.get("https://demo.automationtesting.in/Register.html");
		WebElement dropdownSelectYear = driver.findElement(By.id("yearbox"));
		Select yearDropdown = new Select(dropdownSelectYear);
		yearDropdown.selectByIndex(1);
		yearDropdown.selectByValue("1956");
		yearDropdown.selectByVisibleText("2010");

		// Remember: If we try to select a index, a value, or visible text that
doesn't
```

```
            // exist it will throw an exception
            // It will stop the execution at that line
            /*
             * yearDropdown.selectByVisibleText("A Hundred");
             *
             * List<WebElement> allOptions = yearDropdown.getOptions(); for (WebElement
             * option : allOptions) { System.out.println(option.getText()); }
             */
// Build a logic that will prevent running into a exception when an option is not present
// Use the list to store all available option
            List<WebElement> allOptionsYear = yearDropdown.getOptions();
// Expected visible text - 2023
// Use for loop to iterate over the list and look if there is the option that we are
looking for
// have a boolean that will store the answer
// if we find a match we change the boolean state to true
            boolean isPresent = false;
            for (WebElement option : allOptionsYear) {
                  if (option.getText().equals("A Hundred")) {
                        isPresent = true;

                        break;
                  }
            }

// later by using that boolean we can have a condition
// is the value is present then select it
// otherwise print a hint in the console
            if (isPresent) {
                  yearDropdown.selectByVisibleText("A Hundred");
            } else {
                  System.out.println("The text you are looking for is not present in the
dropdown");

            }
            driver.quit();
      }

}
```

FIND IF ELEMENT IS SELECTED

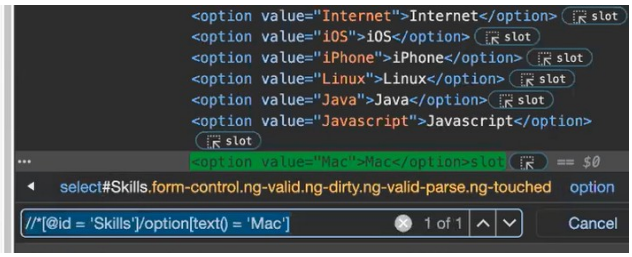
- We can find if certain value is selected from the dropdown via **isSelected()** method.
  - `boolean selectedOrNot = optionElement.isSelected();`

- There is also **getFirstSelectedOption()** method to get first selected item.
  - Then we can extract the text and confirm if text matches our desired option.
  - `Select dropdown = new Select(dropdownElement);`

```
WebElement selectedOption = dropdown.getFirstSelectedOption();
String selectedOptionText = selectedOption.getText();
```

Task:

Find locators by id and xpath of MAC. Then, use getFirstSelectedOption() and isSelect()




```
package module3;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;

public class SelectDropdownPractice2 {
    public static void main(String[] args) throws InterruptedException {

        WebDriver driver = new ChromeDriver();
        driver.get("https://demo.automationtesting.in/Register.html");
        WebElement dropdown = driver.findElement(By.id("Skills"));
        Select skillsDropdown = new Select(dropdown);
        skillsDropdown.selectByVisibleText("Mac");
```

```
        Thread.sleep(5000);
        // How do we programmatically know that an option is selected.
// Option 1: use getFirstSelectedOption() - returns a web element
// getFirstSelectedOption() - will throw an exception if no options are selected
// remember, usually dropdowns have a default option selected
// - in such case it will return you that option
        WebElement selectedOption = skillsDropdown.getFirstSelectedOption();
        System.out.println("Selected Option is: " + selectedOption.getText());

// Option 2: use isSelected() method
        //Mac
        WebElement mySelectedOption = driver.findElement(By.xpath( "//*[@id
='Skills']/option[text()='Mac']"));
        boolean isMacSelected=mySelectedOption.isSelected();
        System.out.println("The Mac option is selected:"+isMacSelected);
        //CSS
        WebElement mySelectedOptionCss = driver.findElement(By.xpath( "//*[@id
='Skills']/option[text()='CSS']"));
        boolean isCSSSelected=mySelectedOptionCss.isSelected();
        System.out.println("The css option is selected:"+isCSSSelected);


        driver.quit();
    }
}
```

```
Aug 07, 2024 9:22:00 PM org.openqa.selenium.devt
WARNING: Unable to find an exact match for CDP v
Selected Option is: Mac
The Mac option is selected:true
The css option is selected:false

Process finished with exit code 0
```

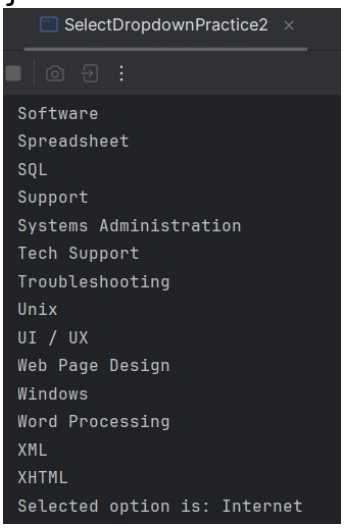
Css is false because it is not selected.

TASK:

//Navigate to http://demo.automationtesting.in/Register.html //Locate the dropdown Skills
//Print all options for Skills dropdown on your console
//Select a skill from Skills dropdown using selectByVisibleText() method
// Print on console selected skill by using getFirstSelectedOption() method

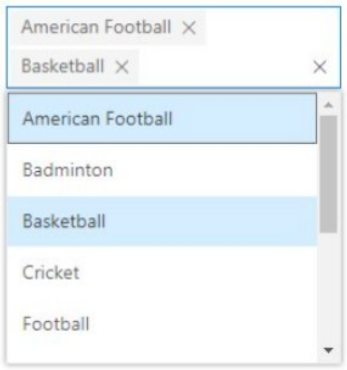
```
package module3;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
```

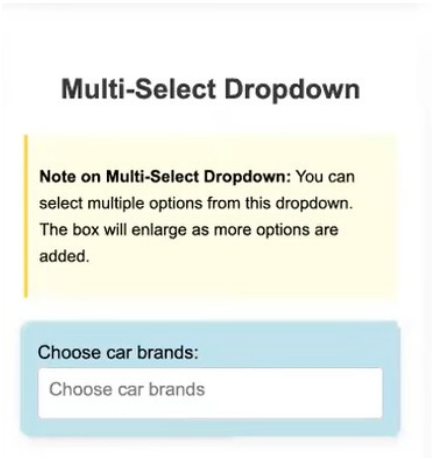
```java
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;

import java.util.List;

public class SelectDropdownPractice2 {
     public static void main(String[] args) throws InterruptedException {

          WebDriver driver = new ChromeDriver();
          driver.get("https://demo.automationtesting.in/Register.html");
          WebElement dropdown = driver.findElement(By.id("Skills"));
          Select skillsDropdown = new Select(dropdown);
          List<WebElement> options = skillsDropdown.getOptions();
          for (WebElement option : options) {
               System.out.println(option.getText());

          }
          skillsDropdown.selectByVisibleText("Internet");
          Thread.sleep(5000);
          WebElement selectedOption =
skillsDropdown.getFirstSelectedOption();
          System.out.println("Selected option is: " +
selectedOption.getText());
          driver.quit();

          driver.quit();
     }
}
```

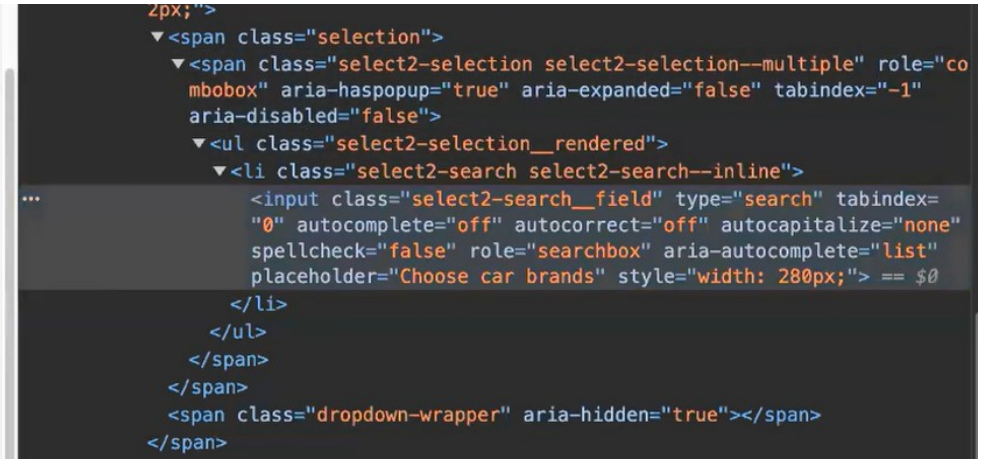


Multiple Choice Dropdowns

- There are cases where application have to let the user choose **multiple values from dropdowns**
- All the methods present in the Select class can be used with multiple value dropdowns as well, as long as they are defined with select tags in HTML.
- Methods specific to Multiple Dropdowns:
  - `myObj.deselectAll();`







Deselect Multiple Values

- Just like we can select values, we can deselect them as well.
- We can deselect using index, visibleText, and value:
  - `dropdown.deselectByIndex(1);`
  - `dropdown.deselectByVisibleText("Rome");`
  - `dropdown.deselectByValue("Rome");`
- In case we want to deselect all the values:
  - `Select dropdown = new Select(dropdownElement);`
    `dropdown.deselectAll();`

```
package module3;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;

import java.util.List;

public class MultiSelectDropdownPractice {

    public static void main(String[] args) throws InterruptedException {
        WebDriver driver = new ChromeDriver();

// With get method we can navigate to url web apps
```

```
// Also, we can navigate to a local html page
        driver.get("file:///Users/dmytrochobotar/IdeaProjects/SeleniumC13/resources/uiPages/m3lesson1/index.html");
        Thread.sleep(5000);
// Interacting with multi select dropdown
        WebElement dropdown = driver.findElement(By.id("multiSelect"));
// Create a select object
        Select multiSelectDropdown = new Select(dropdown);
        multiSelectDropdown.selectByIndex(0);
        Thread.sleep(5000);
        multiSelectDropdown.selectByVisibleText("Audi");
// Select object provides you couple of methods that are intended to use on multi select dropdowns
// If you try to use them on simple select dropdown it will throw an exception
// all of the below methods will work with multi select dropdowns:
        /*
        multiSelectDropdown.deselectAll();
        multiSelectDropdown.deselectByVisibleText();
        multiSelectDropdown.deselectByIndex();
        multiSelectDropdown.deselectByValue();
        */

        List<WebElement> selectedOptions = multiSelectDropdown.getAllSelectedOptions();
        for (WebElement option : selectedOptions) {
            System.out.println(option.getText());
        }
        //To deselect all
        multiSelectDropdown.deselectAll();
```

```
    driver.quit();
}}
```

Custom Dropdowns

- There are many ways of creating dropdowns, and some of them are not using regular select dropdown format.
- These are custom dropdowns. You can use following different methods to select values from those dropdowns.
- Depending on the dropdown you have, you will choose which strategy to use.

We can select the dropdown values in few ways:

- Click method
- Actions class click methods
- Find element method
- Sendkeys method
- Javascript method

DROPDOWN is located differently:

The developpers develop the options in different way, they use SELECT or CUSTOM-DROPDOWN, each case needs to be located by XPATH in different approach.

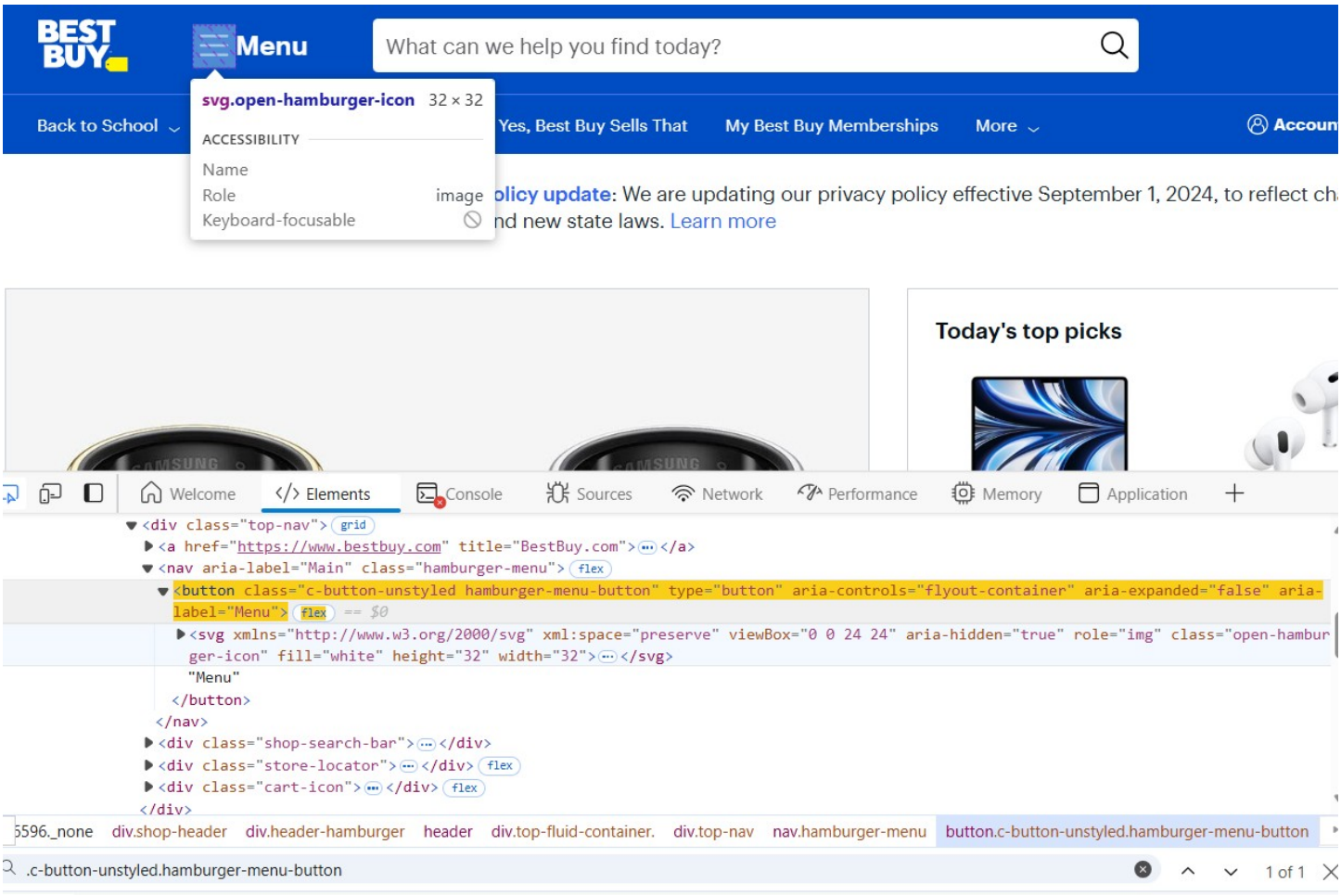
What to do if you don’t have : SELECT ?

Example : Hamburger Menu

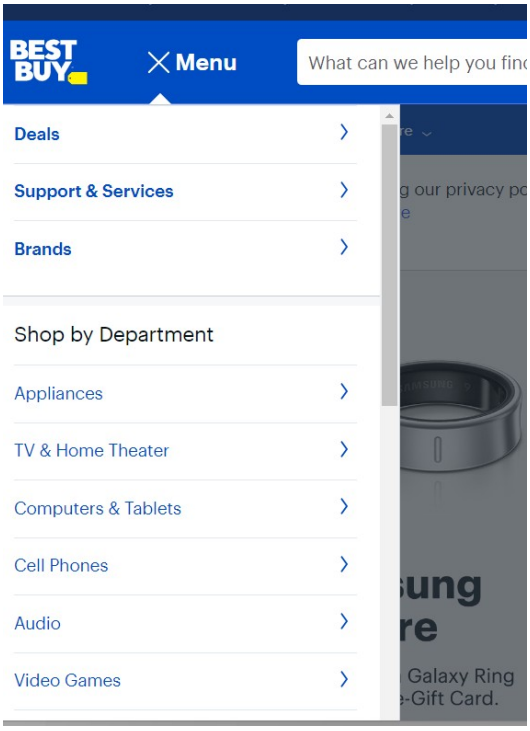
TasK

We use CSS Locator : .c-button-unstyled.hamburger-menu-button, then menu is selected.

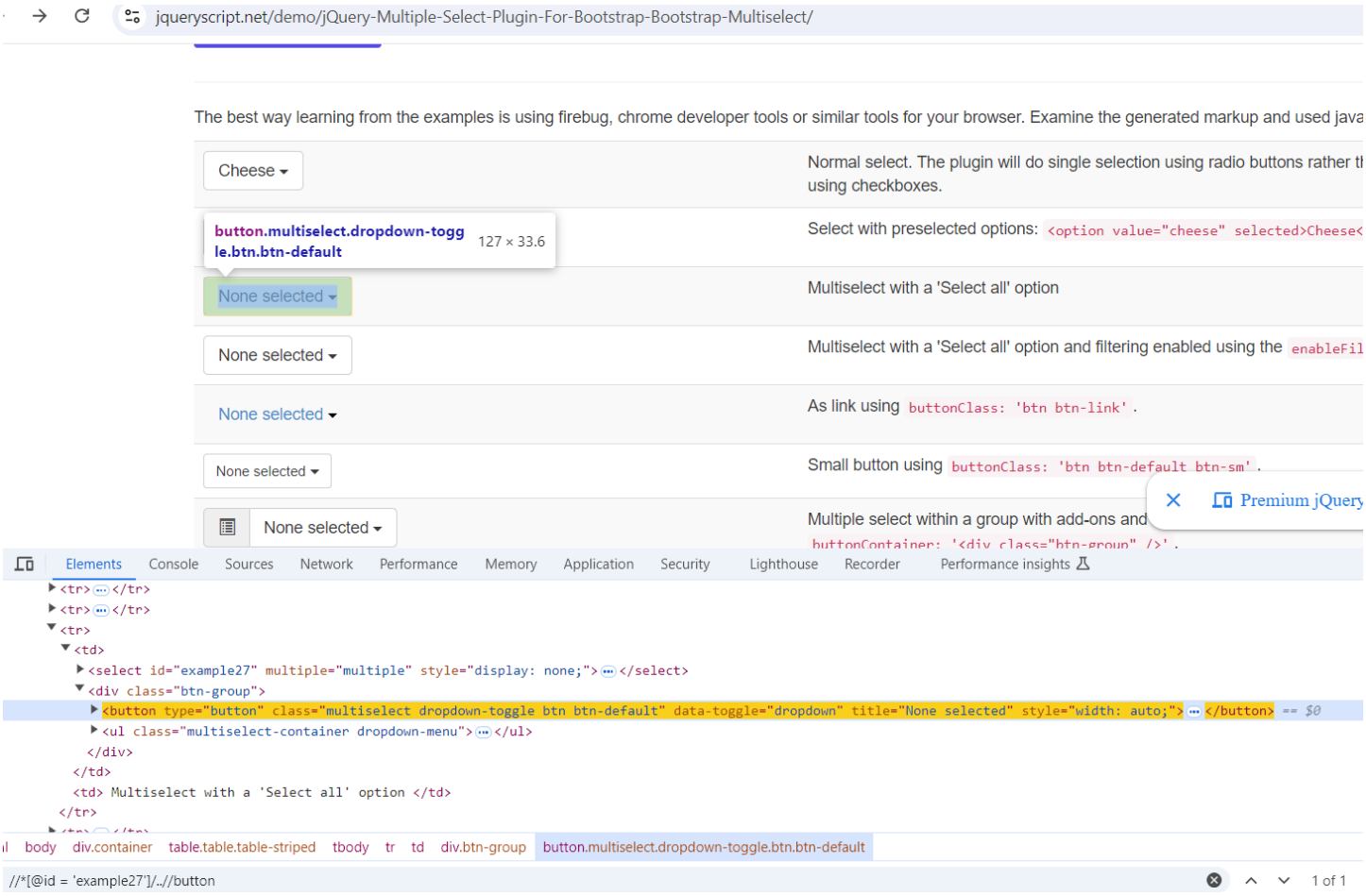
Choose BRANDS, then choose Apple.


BEST BUY
Menu
What can we help you find today?
svg.open-hamburger-icon 32 × 32
ACCESSIBILITY
Name
Role image
Keyboard-focusable
Back to School
Yes, Best Buy Sells That
My Best Buy Memberships
More
Account
olicy update: We are updating our privacy policy effective September 1, 2024, to reflect ch
nd new state laws. Learn more
Today's top picks
Welcome
Elements
Console
Sources
Network
Performance
Memory
Application
 grid
...
<nav aria-label="Main" class="hamburger-menu"> flex
<button class="c-button-unstyled hamburger-menu-button" type="button" aria-controls="flyout-container" aria-expanded="false" aria-label="Menu"> flex == $0
<svg xmlns="http://www.w3.org/2000/svg" xml:space="preserve" viewBox="0 0 24 24" aria-hidden="true" role="img" class="open-hamburger-icon" fill="white" height="32" width="32">...</svg>
"Menu"
</button>
</nav>
...
... flex
... flex

5596._none
div.shop-header
div.header-hamburger
header
div.top-fluid-container.
div.top-nav
nav.hamburger-menu
button.c-button-unstyled.hamburger-menu-button
.c-button-unstyled.hamburger-menu-button
1 of 1


BEST BUY
Menu
What can we help you find
Deals
Support & Services
Brands
Shop by Department
Appliances
TV & Home Theater
Computers & Tablets
Cell Phones
Audio
Video Games
Galaxy Ring
Gift Card.

```java
// We need to interact with Menu Dropdown
// This dropdown doesn't have a select tag
// So, we will need a custom strategy
// For example the menu dropdown activates the options when we click on it
// So the strategy will be to locate the dropdown button, click on it and then
choose the option with a custom locator
package module3;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class CustomDropdownPractice {
    public static void main(String[] args) throws InterruptedException {
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.bestbuy.com/");
//Select menu Hamburger
        WebElement menuHamburgerDropdown = driver.findElement(By.cssSelector(".hamburger-
menu-button"));
        menuHamburgerDropdown.click();
//Select Brands from the menu
        WebElement brandsOption = driver.findElement(By.xpath(  "//button[text() =
'Brands']"));
        brandsOption.click();
        Thread.sleep( 5000);
//Select Apple from the menu

        WebElement appleBrand = driver.findElement(By.xpath( "//ul[@class = 'hamburger-
menu-flyout-list hamburger-menu-flyout-sidecar-list']//a[text()='Apple']"));
        appleBrand.click();
        Thread.sleep( 5000);
        driver.quit();
}}
```



Explaining the xpath ://*[@id = 'example27']/..//button

Go to select :id=”example27”

Then, Go back to <td> using : /..

This moves up one level in the DOM (Document Object Model) hierarchy, selecting the parent element of the element with id="example27".

Jump to all <button> using : **//button**

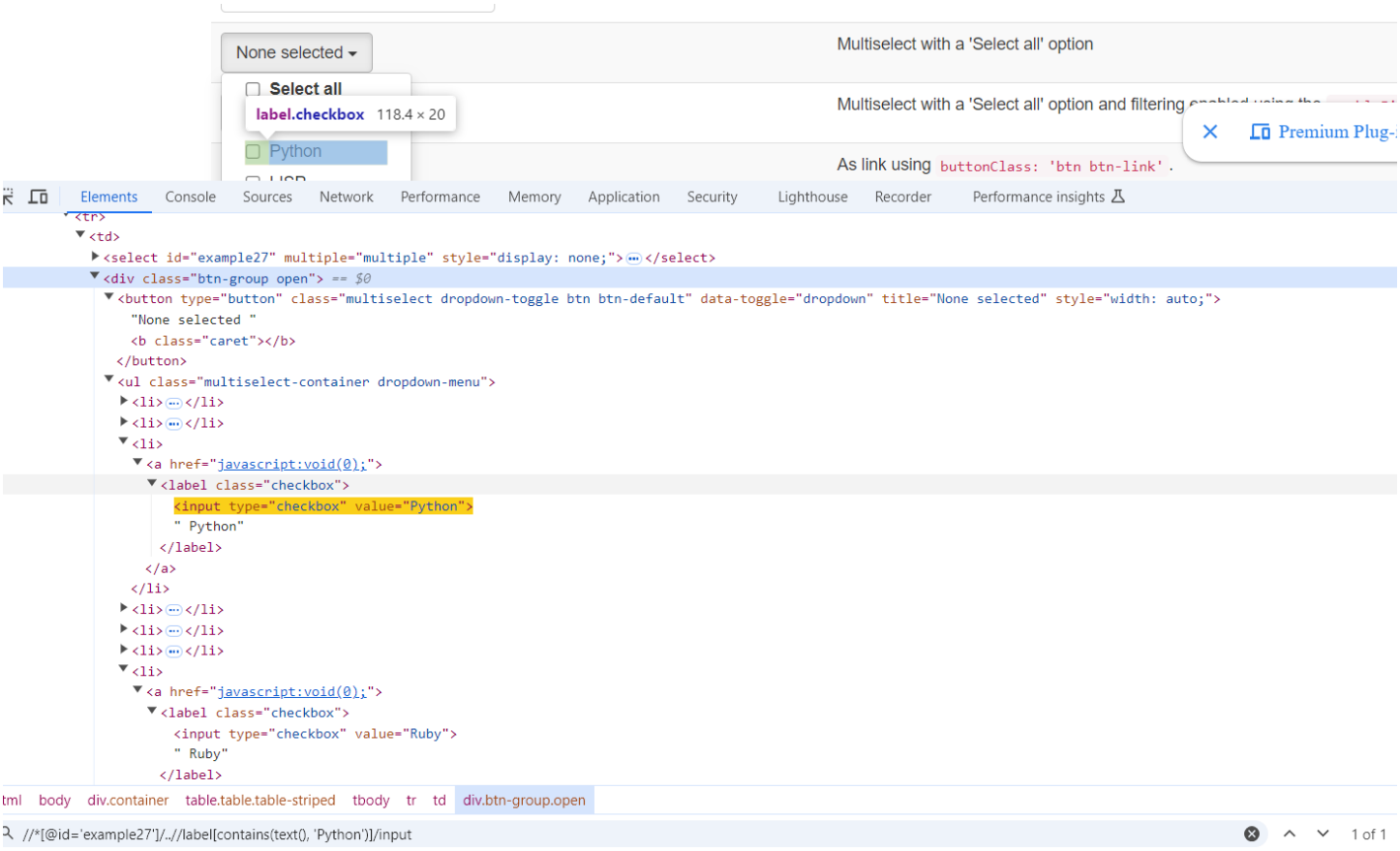
This selects all <button> elements that are descendants of the parent element found in the previous step.

```
package module3;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
public class CustomDropdownTask {
    public static void main(String[] args) throws InterruptedException {
        WebDriver driver = new ChromeDriver();

driver.get("https://www.jqueryscript.net/demo/jQuery-Multiple-Select-Plugin-For-Bootstrap-Bootstrap-Multiselect/");
Thread.sleep( 3000);
```

```
WebElement thirdDropdown = driver.findElement(By.xpath(  "//*[@id =
'example27']/..//button"));
        thirdDropdown.click();
Thread.sleep( 3000);

driver.quit();
}}
```

Task

```
/*
      * Navigate to https://www.jqueryscript.net/demo/jQuery-Multiple-Select-
Plugin-For-Bootstrap
      * -Bootstrap-Multiselect/ Select "Python" and "Ruby" from third dropdown Use
      * ul/li tags for this task
            */
```




```
package module3;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
```

```
public class CustomDropdownTask {
	public static void main(String[] args) throws InterruptedException {
		WebDriver driver = new ChromeDriver();


		driver.get(
				"https://www.jqueryscript.net/demo/jQuery-Multiple-Select-Plugin-For-Bootstrap-Bootstrap-Multiselect/");
		Thread.sleep(3000);
//go to
		WebElement thirdDropdown = driver.findElement(By.xpath("//*[@id = 'example27']/..//button"));
		thirdDropdown.click();
		Thread.sleep(3000);
// TODO - Improve this by creating a reusable method that takes as a parameter a string
// Then clicks on that option, additionally you can validate if the option is present in the dropdown
// Checkbox the Python option

		driver.findElement(By.xpath("//*[@id='example27']/..//label[contains(text(), 'Python')]/input")).click();
		Thread.sleep(3000);
		driver.findElement(By.xpath("//*[@id='example27' ]/..//label[contains(text(), 'Ruby')]/input")).click();
		Thread.sleep(3000);
		driver.quit();
		driver.quit();
	}
}
```

XPath Expression:

`//*[@id='example27']/..//label[contains(text(), 'Python')]/input`

Breakdown of the XPath:

1. **`//*[@id='example27']`**:
   - `*`: The asterisk (*) is a wildcard that selects any element type.
   - **`[@id='example27']`**: This filters the selection to elements that have an id attribute with the value 'example27'.
   - This part of the XPath selects the element with the id of "example27". The element could be any HTML tag (e.g., , , etc.).
2. **`/..`**:
   - This moves one level up the DOM tree, selecting the parent element of the element with id='example27'.
   - Essentially, it navigates from the child element (with id='example27') to its parent element.

3. **//label[contains(text(), 'Python')]**:
    - **//**: This double forward slash selects all descendant elements (children, grandchildren, etc.) of the parent element.
    - **label**: This specifies that the search is for <label> elements.
    - **[contains(text(), 'Python')]**: This filters the <label> elements to those whose text content contains the string 'Python'.
    - This part of the XPath finds a <label> element within the parent element that contains the text "Python".
4. **/input**:
    - **/input**: This selects the <input> element that is a child of the <label> element found in the previous step.
    - The <input> element is typically used for form inputs, such as checkboxes, radio buttons, or text fields.